SYM-AM-23-086

Excerpt from the Proceedings of the Twentieth Annual Acquisition Research Symposium

Acquisition Research:
Creating Synergy for Informed Change

May 10–11, 2023

Published: April 30, 2023

Approved for public release; distribution is unlimited.

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School

Optimizing Operations and Logistics Support Using Opus Evo

Gustaf Solveling—is the principal analyst and senior software developer at Systecon North America with more than 10 years of analytics and software development experience. Solveling has 10 years of logistics support system optimization experience as consultant and decision support system developer and has served as a project manager and technical leader for software development projects, integration projects, and data science projects. He has a PhD in Industrial Engineering and a Master of Science in Operations Research from Georgia Institute of Technology and a Master of Science in Communication and Transportation Engineering from Linkoping University. [Gustaf.Solveling@systecon.us]

John Verbanick—is a senior analyst with three years' experience at Systecon North America. He served 15 years as a Weapon Systems Officer in the United States Air Force. After leaving the military Verbanick went to work as a Manufacturing Engineer supervisor with Honeywell Aerospace, working on manufacturing process control, supply chain improvements, and process improvements based off Six Sigma analysis. He has a Bachelor of Science in History from the United States Air Force Academy and a Master of Military Operational Arts and Science from the Air Command and Staff College. [John.Verbanick@systecon.us]

Abstract

Understanding the interplay between the reliability and maintainability of a fleet of complex systems, the logistics support organization, and the operational scenario, is vital from a short-term tactical perspective, as well as a strategic long-term Life Cycle Cost (LCC) perspective, as each of these areas has a direct impact on one another. A common method to analyze and evaluate the performance of the overall scenario, as well as getting insights into problem areas, bottlenecks, and to perform analysis-of-alternatives is to use discrete event simulation.

In this paper we present a methodology to extend a discrete event simulation tool with inherent optimization capabilities. Using established heuristic optimization techniques, we perform simulation driven optimization that optimizes parameters in the modeled scenario. Optimized parameters typically include:

- sparing strategies such as inventory levels and locations
- resource quantities and location
- deployed system quantities to fulfill mission requirements
- scheduled maintenance times.
- transportation and resource schedules.

A case study is presented that utilizes Opus Evo, an application that extends the commercial off the shelf Opus Suite with capability to perform heuristic optimization using simulation.

Keywords: Heuristic Optimization, Tactical Logistic Planning

Introduction

Predicting and optimizing mission capability and readiness for a system requires knowledge and data in a range of areas that each have a direct impact on the outcome. The reliability of the components making up the system, the maintainability of the system, the responsiveness of the support organization, and the operational tempo are all factors that contribute to, or inhibit, readiness. Furthermore, mission capability and readiness are always associated with a cost and understating the relationship between cost and readiness is important, especially when optimizing readiness given budget constraints.

To represent the modeled scenario it is important to have a suitable domain model. An appropriate domain model simplifies data entry, promotes the understanding of the model, is compatabile with established standards, etcetera, but in the end, the data in the domain model is used for analytics which provides insights and recommendations. In the Domain Model section we present characteristics of a domain model that can be used for evaluation and optimization of system availability and mission readiness.

Opus Suite is a suite of software applications that is used for predictive analytics of complex technical systems together with its operational characteristics and its logistics support network. One of the core applications in the suite is SIMLOX, a discrete event simulation tool for predicting mission performance and readiness over time. The objective of a simulation tool is typically to evaluate the behavior of a system given stochastic parameters and stochastic dynamics, where the analysis is limited to simulation of one scenario at a time.

In this paper we present a method that enables *optimization* of any entity in the domain model. Enabling optimization makes it possible to not only analyze and evaluate the performance of the overall scenario, but also identify improvements in a systematic way without a manual "trial and error" process. The method is based on evolutionary algorithms, and we show how any numeric data element in the domain model can be optimized. In the next section, the domain model is introduced with examples on what can be optimized using the proposed methods. Following that, the application in which the method has been implemented is presented together with an algorithm description. In the last section we present a case study where the application, which goes by the name of *Opus Evo*, has been utilized to optimize a pack-up kit for deployed operations of an aircraft system. The methodology presented in this paper is general in nature and does not depend on a specific set of tools or applications. For the proof of concept and for the case study, the Opus Suite has been utilized to provide a domain model and optimization evaluator.

Domain Model

A domain model to support mission readiness and system availability optimization requires representation of data in a number of categories. Examples of the categories are:

- Product breakdown
- Reliability
- Task
- Corrective and preventive maintenance event
- Maintenance capabilities
- Operation profiles
- Functional breakdown/reliability block
- Mission characteristics
- Inventory
- Resources

Within each category there are typically several entities with associated attributes. In general attributes can be of any data type, but for use with the evolutionary algorithm presented in the next section, attributes to be optimized need to be numerical values. An example of an entity that can be optimized is seen in Table 1, where a typical objective is to maximize readiness by determining inventory levels and locations subject to a budget constraint.

Table 1. Entity in Domain Model

Entity:	Inventory Level
Key:	Item Identifier
Key:	Location
Attribute:	Nominal Stock Level
Attribute:	Item Cost
Attribute:	Storage Cost

As will be seen in subsequent sections, the proposed methodology enables optimization of any entity in the domain model. Examples of scenarios that can be optimized are listed below. In all examples below, the objective is to maximize mission capability and readiness.

- Maximize mission capability by optimizing inventory levels subject to budget constraints.
- Maximize mission capability by optimizing resource quantities and locations subject to budget constraints.
- Maximize mission capability by optimizing the mix of inventory vs. resources given budget constraints.
- Minimize deployed system quantities while achieving a specified mission capability level.
- Maximize mission capability by optimizing the time of maintenance given specified maintenance windows.

Technical Solution

Evolutionary Algorithms

Evolutionary algorithms are heuristic optimization algorithms inspired by processes in nature. The algorithms are applicable to many optimization problems, as the algorithms typically only require evaluation of the objective function, often referred to as the fitness function, to determine the quality of a single solution for the problem at hand. Although the principles of all evolutionary algorithms are the same, the methods performed in the general steps may differ, which creates several types of evolutionary algorithms. Due to the nature of the optimization problems represented in the domain model, the evolutionary algorithm type that have been tested and implemented is *differential evolution.* Differential evolution is especially suited for problems where the variables to be optimized are numerical values (in contrast to binary values typically used for genetic algorithms).

The basic algorithm steps of all evolutionary algorithms are (Simon, 2013):

1. Randomly generate the starting sample set
2. Evaluate the fitness function of all samples
3. Select the best samples to keep for reproduction (parents)
4. Combine and create new samples from the parents (offspring)
5. Replace least fit samples with new offspring (survival of the fittest)
6. If termination criterion is not reached, go back to step 2, otherwise terminate, and return the sample with the best fitness as the solution.

The calculations specific to differential evolution take place in steps 4 and 5 in the algorithm above, where new samples are generated according to the following procedure (Price et al., 2005):

1. For each sample, $\bar{x} \in \mathbb{R}^n$, in the population, create a new sample, $\bar{y} \in \mathbb{R}^n$, according to:
    a. Select three samples from the population, distinct from $\bar{x}$ and from each other. Call these $\bar{a}$, $\bar{b}$, and $\bar{c}$.
    b. Determine a subset J $\sqsubseteq \{1, \dots, n\}$, such that $|J| \geq 1$ (at least one in the vector dimension will change).
    c. For each j$\in J$ let $y_j = a_j + F \times (b_j - c_j)$
2. If the fitness function applied to $\bar{y}$ gives a better value than for $\bar{x}$, replace $\bar{x}$ with $\bar{y}$ in the set of samples.

The subset J in step 1b is determined using a constant called *crossover probability* and the mutation in step 1c uses a constant *F* called *crossover factor*. For further details on the differential evolution algorithm, see Price et al. (2005).

The standard differential evolution algorithm is extended with a taboo list so that samples whose fitness function value is already know do not need to be evaluated a second time.

Opus Evo

The application in which the proposed methodology has been implemented is referred to as *Opus Evo*. Opus Evo is an application within Opus Suite that enables optimization of any attribute, or combination of attributes, within the domain model. The optimization is performed using the differential evolution algorithm presented above. The application is made up of several components, which are listed below. For each component, a general description is provided together with additional details that are particular to Opus Evo.

The core differential evolution algorithm: Given a representation of the problem in form of a sample and a fitness function the algorithm will find a solution whose fitness function value is not worse than that of any other solution discovered. A global optimum cannot be guaranteed for evolutionary algorithms. Note that the algorithm itself does not need to know anything about the problem being optimized, other than the representation of a sample and what fitness function to use. The steps of the algorithm are described in The Evolutionary Algorithms section.

A domain representation, problem data, and data storage: This is where the problem that is being solved is modeled. A general domain model is described in the Domain Model section. In Opus Evo, the existing Opus Suite domain model and data storage is leveraged. Thus, a model instance that already exists in Opus Suite can be optimized in Opus Evo.

A domain to vector mapping: The evolutionary algorithm requires the optimized attributes in a vector representation, but the algorithm does not need to know what the values in the vector represent. To achieve this, it is necessary to have a mapping from entity attributes in the domain model to the internal vector representation. In Opus Evo, the mapping of entity attributes to the vector representation is provided through a text file specified by the user. In the variable declaration, bounds on the variables can be provided. The mapping is necessary to translate the two representations, e.g., when evaluating the fitness function or when to present the best solution to the end user.

A fitness function: The purpose of the fitness function is to determine the fitness, or quality, of a given solution. In the context of the evolutionary algorithm, a fitness function is a black box, which takes a vector of values as input, and return a single- or multi-dimension objective value as output. Opus Evo uses the simulation tool SIMLOX for fitness evaluation. The optimization algorithm can use any fitness function, but using SIMLOX has several advantages:

- SIMLOX is an evaluator/function that already can handle all aspects of the domain model. Thus, the same fitness function can be used regardless of what entities being optimized.
- The result of SIMLOX is mission capability or system availability, which is typically the desired objective for an optimization in this domain.
- Discrete event simulation works well with distributed computing, where evaluation of individual samples can be distributed. Furthermore, replications within a simulation run can be distributed and processed in parallel.

The fitness function can support multi-dimensional objectives, so it is possible to simultaneously optimize different dimensions (e.g., cost and readiness). However, it has been observed that the algorithm progress to an optimal solution quicker if a single objective is considered.

A computing resource orchestrator. Evaluation of samples during an iteration of the algorithm can be distributed and performed in parallel, as each sample is independent from all other samples. Thus, the performance of the algorithm scales linearly with the number of processors available. Opus Evo includes support for distributed processing using the Message Passing Interface (MPI; Microsoft, 2022). For the case study presented in the next section, a cluster with a total of 70 logical processors distributed over two servers was utilized.

The domain representation and the fitness function are presented in the context of Opus Suite, but the application is general and any domain representation that meets the criteria in the Domain Model section can be used, as well as any fitness function that is able to evaluate a solution can be used.

Case Study Air Force Deployment

The case study being presented deals with looking at optimizing spare parts with requirements outside the typical cost vs. availability tradeoff. This problem required transporting equipment to a new location along with all the spare parts for that equipment. However, this location has limited access and constraints in space available to set up the proposed operation. Given these parameters the problem further required that the equipment to be operational over a 20-day period, given the requirement that the availability rate exceeds 98% of the desired operating window.

For this problem, an initial package of spares had already been created with an operational design to last for a 20-day period while providing a high availability rate, but these off-the-shelf packages are based on an equipment usage rate twice that of what is expected over this proposed excursion. The packages also assume that no restrictions exist based on transportation and storage space for the required parts. A typical modeling approach would provide an optimized solution based on a cost vs. availability curve. With this solution curve, further analysis would be required to find and isolate the numerous solutions that do not meet an optimal curve point. With this problem, the introduction of a new factor to analyze becomes necessary, in this problem that constraint is the dimensional data of all the spare parts that are modeled. Typically, the modeling solution focuses solely on maintenance significant items (MSIs), or those items which have been determined to be necessary to keep the equipment in a maintenance up and operational status based off historical failure rate analysis of all the components of the equipment.

Addressing this problem within the newly defined constraints, one must start with a basic cost vs. availability model of the system. SIMLOX is designed to perform this analysis and an existing model of the equipment existed that could serve as a baseline for the new solution. From this baseline one can take the provided solution, utilize the Opus Evo application, and run the model with the dimensional restrictions to develop a solution that becomes acceptable to the requirements and limitations that were laid out.

The case study involved the movement of 24 pieces of equipment with each typically requiring a full transportation unit to move and have available. The transportation cost of 24 transport units was determined to be too much and unacceptable to the problem's end state needs. The problem had another complication in that the new location would not be able to accommodate the footprint of equipment that these transportation units would bring with them. As such, a determination and innovative approach is required to achieve operational success which would provide capability while reducing the dimensional limitations that are typical given the movement of equipment.

The requirement that the equipment have the capability to operate every day for the 20-day period with four systems in operation at any given time while those not in operation requiring a two-hour pre-operating period and two-hour post maintenance period provided an added challenge to the problem set. A total of 1,200 hours (about 20 days) of actual operational coverage provided by the equipment is necessitated to achieve operational success. These restrictions are half of what the off-the shelf spares package provides, so the overall question became what is required to achieve success while reducing the maintenance and storage footprint to the max extent possible?

Typically, if one were to create and run this model, it could be accomplished through traditional modeling means. However, this approach adds a cost associated which is the time to compute. As discussed, to find the best fit solution you would first optimize the model and then take that solution as a baseline. Once this baseline is established you can proceed in one of two ways. The first solution requires a degree of time through trial-and-error, running iterations with slightly different data points to narrow down the result into one that fits the operational requirements. This requires looking at the data from each iteration and performing calculations on the results outside modeling software. The impact of this is after every modeling run the modeler is required to choose outcomes that bring the solution in line with requirements. The added time of this becomes excessive to the modeling process. As one can imagine, this considerable time cost to run the iterations and analyze the results to find the best solution for the problem makes meeting a shortened timeline unacceptable to requirements.

The second option builds off the first, but it is to run two additional iterations to the baseline model and then narrow down the solution set utilizing interpolation of the results to find a working solution, this would be useful to save time in a smaller model, however the size and scope of this problem makes this prohibitive in application, requiring the modeler to make assumptions that may or may not be realistic to the problem at hand. This approach, like the previous requires taking the results from three runs and interpolating the results to find an acceptable solution. This approach brings artificiality into the modeling process and does not guarantee an optimized result.

For this problem, the baseline model requires approximately eight hours of computing time, with each follow-up iteration requiring the same time cost. This requires the modeler to spend at a minimum 24 hours, and possibly days of computational time looking at different iterations and factors. Computational time alone is an issue; however, the time required to analyze the results and determine a workable solution could take additional weeks to sort. The time sink that these approaches bring make these two solutions unworkable. If time is not a factor, one can brute force and with some luck achieve an acceptable solution in a matter of weeks/months. However, for this problem the requirement is a working solution within days. The timeline for this problem is thus prohibited in the traditional modeling capability and at best using these means one can attain a rough solution and perform a post event analysis between our solution and what they utilized.

With Opus Evo one can run numerous iterations over a brief period with the new constraints utilizing heuristic simulation on the baseline model. The new tool allows one to use the baseline model and then utilize machine learning and the power of heuristic simulation provided through Opus Evo. This tool allows different iterations of the baseline model to be run concurrently, utilizing a third factor outside the traditional cost vs. availability. Given an appropriate expanded data set, one can utilize this approach and optimize a model based off any third factor. With the right data this software permits the modeler to develop an optimized solution in a fraction of the time that it would take to perform the task manually. One can now utilize this capability for the problem of creating a solution meeting their requirements in both system availability, reduced maintenance, and storage footprint in a shortened timeline. To validate the solution in both effectiveness and time indicators, the results were compared between the modeled solution and the planned solution the problem utilized to determine if the new Opus Evo software indeed could optimize the problem while meeting the operational requirements.

When we compared the modeled operational results with those provided as the user solution, the modeled results produced a stark contrast in system availability, number of parts required, and cost associated with the scenario. The top chart in Figure 1 displays that utilizing Opus Evo the modeled solution ensured a greater number of systems available and in operations throughout the entire 20-day (480-hour) window whereas the solution provided Figure 1 bottom chart shows the equipment becoming unsustainable at the 15-day(360-hour) point of their operations.

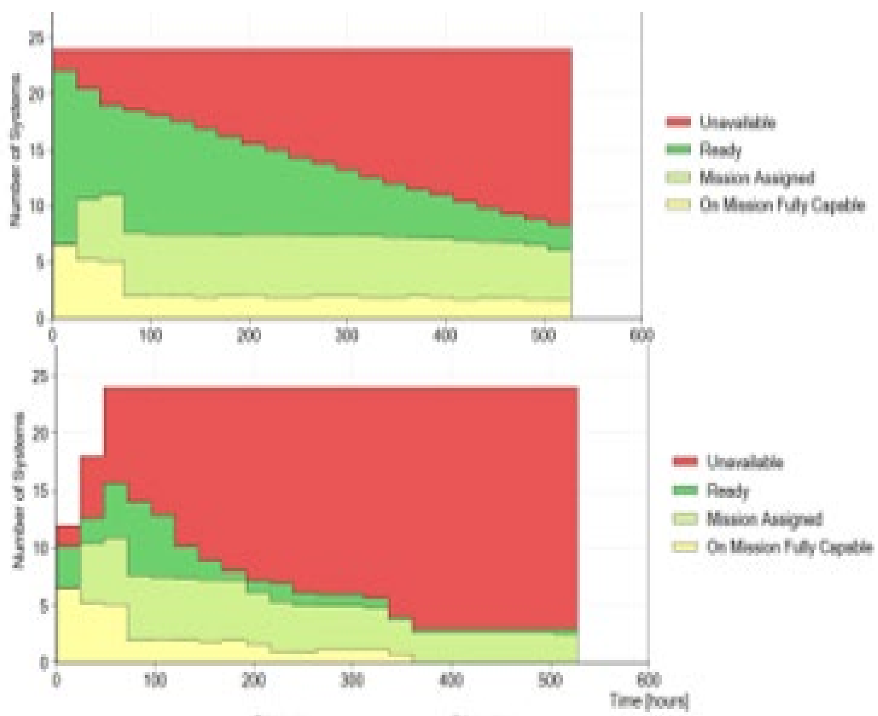

Figure 1. Simulation Results Opus Evo on the Top; Customer Solution on the Bottom

The contrast in solutions shows a distinct advantage to utilizing Opus Evo. The modeled solution on the top shows that system availability was maintained at eight systems at any given time throughout the entire 20-day period. Whereas the solution we were given drops to three systems available at the 15-day point. This savings in capability alone shows that in modeling through Opus Evo, we can optimize the availability of the systems and meet the problem requirements of four systems in operation at any given time, while also enabling an added surge capacity in usage. The savings in this aspect alone from the modeled solution over the provided solution is enough to prove out the software capabilities.

To further prove the ability of Opus Evo, the ability to meet the problem requirements, in post action analysis it was determined that the problem solution planned to under deploy 55 MSI parts as well as over deployed by 296 parts over what the modeled solution provided. In total by utilizing Opus Evo, it was determined that one could meet the requirements with a 38% reduction in parts, a 56% reduction in cost, and a 50% increase in equipment availability.

References

Microsoft. (2022). *Microsoft MPI*. https://docs.microsoft.com/en-us/message-passing-interface/microsoft-mpi

Price, K., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution: A Practical approach to global optimization*. Springer.

Simon, D. (2013). *Evolutionary optimization algorithms*, Wiley.

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net