A multi-criteria policy set optimisation framework for large-scale simulation models

David Myers*

US Air Force Research Laboratory,
525 Brooks Road, Rome, NY 13441, USA
Email: David.Myers.35@us.af.mil
*Corresponding author

Mark H. Karwan

Department of Industrial and Systems Engineering,
University at Buffalo (SUNY),
305 Bell Hall, Buffalo, NY 14260, USA
Email: mkarwan@buffalo.edu

Abstract: Simulation modelling is an analysis approach utilised in nearly every domain for analysis of large and complex systems. Synchronous data flow (SDF) is used to model systems whose data does not follow the predetermined global schedule of discrete event simulation modelling techniques. A policy set optimisation (PSO) problem for any simulation model is the selection of a small set of controllable inputs for manipulation by a decision maker (DM) in order to achieve a desired goal. This is a multi-criteria decision making problem and a large-scale SDF simulation model creates a complex mathematical model for solution. Our PSO framework and associated procedure aims to generate the policies that will provide an estimation of the Pareto optimal solutions for the simulation model using only pre-processed model sampling. Our solution methodologies for the PSO problem aims to minimise the computation time required from the point at which a DM selects the outcomes of interest, to when they receive solution policies to choose from. This paper provides a sample problem and a discussion about the quality of the solution found.

Keywords: multi-criteria decision-making; MCDM; policy set optimisation; large-scale simulation models; synchronous data flow; mathematical modelling.

Reference to this paper should be made as follows: Myers, D. and Karwan, M. (2015) 'A multi-criteria policy set optimisation framework for large-scale simulation models', *Int. J. Applied Nonlinear Science*, Vol. 2, Nos. 1/2, pp.49–74.

Biographical notes: David Myers is a Research Engineer for the US Air Force Research Laboratory (Information Directorate). He received his PhD in Operations Research in the Department of Industrial and Systems Engineering from the University of Buffalo, The State University of New York in September 2013. He has been involved as a researcher in several projects funded by the Department of Defense including work in the domains of unmanned aerial vehicles, large-scale simulation models, multi-criteria decision-making and cyber defence.

Mark H. Karwan is a Praxair Professor in Operations Research and a SUNY Distinguished Teaching Professor, has served 38 years in the Department of Industrial and Systems Engineering at the University at Buffalo. His broad expertise is in mathematical programming modelling and algorithmic development. He has 30 PhD students and 37 MS students have been guided in algorithmic development in integer programming, multiple criteria decision making and mixed areas such as integer/nonlinear or integer/multi-criteria. He has 90+ publications show diverse application areas such as logistics, production planning under real time pricing, capacitated lot-sizing, hazardous waste routing and security, military path planning and analytics. Funding has come from NSF, ONR, AFRL and industry. His industry consulting experience focuses in two fields: 1) the industrial gas industry in all areas of production planning, routing, forecasting, energy use planning, cost/pricing; 2) defence agencies and contractors in military operations research focused on logistics and dynamic resource allocation.

1 Introduction

Simulation modelling is a common approach for understanding and analysing large and complex systems. Traditional domains such as manufacturing, much less traditional domains such as the finance industry, and everything in between utilise simulation in order to create decision support tools for an analyst or decision maker (DM). A typical decision for a DM in all domains contains more than one single objective. Multi-criteria decision-making (MCDM) considers more than one, often times conflicting objectives simultaneously.

The goal of any MCDM process is to help a DM consider multiple objectives simultaneously as well as find a solution that pleases the DM the most. There are many desirable characteristics of MCDM processes. The process should generate Pareto optimal solutions reliably, as well as provide an overview of the set of Pareto optimal solutions to the DM. The method should also not require too much time or demanding information from the DM. It should also support the DM in finding the most preferred solution.

Our PSO framework embraces MCDM in the selection of a policy that optimises multiple outputs of interest simultaneously. A policy is a set of actions taken on the inputs of the simulation model. Our work utilises mathematical modelling to optimise the selection of these policies. The target of this work is a large-scale simulation model that contains many controllable inputs. Our goal for this application is the minimisation of computation time required to generate reasonable policies for a DM. A reasonable policy is a policy set that is implementable by a DM (not too many inputs to manoeuvre in order to implement). Unfortunately, target applications of a size that properly stress our mathematical models solution capabilities are proprietary in nature. We present a simple sample model and results associated with that sample model to show the e effectiveness of finding a set of good policies for a DMs consideration. We also present some computational results for one approach from large-scale data to show solution capability with a model of larger size.

The contribution of this research is the generation of policies, solely based on pre-processed simulation runs, in large-scale simulation models that consider a DM and their multi-criteria interests. We utilise two solution approaches in order to generate these policies in a computationally efficient manner. We compare two solution approaches and visualise results for the DM in order to select their preferred solution policy.

The rest of this paper has the following layout. A review of the literature is in Section 2. Section 3 describes the PSO framework. The first solution approach is the model decomposition approach (MDA) described in Section 4. The second solution approach is the model sampling approach (MSA) in Section 5. Section 5 also contains the computational results for large-scale data to show solution capability. A description of the PSO system is in Section 6. The visualisation approach utilised by the PSO system is in Section 7.

2 Literature review

In a MCDM problem, there are two main tasks. The first task is the optimisation task. The purpose of the optimisation task is to find Pareto optimal solutions, also known as the set of non-dominated solutions. A non-dominated solution is any solution for which there exists no other feasible solution that obtains a 'better' objective value in all objectives. 'Better' refers to a greater or equal value in terms of maximisation or lesser or equal value in terms of minimisation. Despite the fact that there are multiple Pareto optimal solutions, it is typical that the DM chooses only one of these solutions. Our framework and associated procedure aims to generate the policies that will provide an estimation of these Pareto optimal solutions for a simulation model using only pre-processed model sampling.

The second task in a MCDM problem is the decision-making task for choosing the single most preferred solution among this set. While the focus of this paper is the optimisation task of the PSO process, the framework considers the decision-making task throughout. The DM in any application of large-scale simulation modelling applies unique constraints to the PSO problem at hand that make this problem different from the typical multi-objective optimisation problem examined in the literature.

Dyer et al. (1992) and Wallenius et al. (2008) provide good reviews of the history of MCDM and multi-attribute utility theory (MAUT). As predicted in Wallenius et al. (2008) there has been quite a bit of recent literature on evolutionary multi-objective optimisation (EMO) method. Coello-Coello et al. (2006), Branke et al. (2008), Deb et al. (2002), Aittokoski and Miettinen (2010), and Eskandari et al. (2005) each develop new evolutionary optimisation algorithms or expand on past algorithms for use in MCDM problems. *Multi-objective optimisation using evolutionary algorithms* (Deb, 2001), is a good resource for the utilisation of these evolutionary algorithms in a multi-objective setting. EMO algorithms in general require many simulation runs after a DM provides preference information in order to generate efficient solutions to the problem. Obviously, a large number of simulation runs cannot occur in all domains. The DM must select outputs of interest, and in a timely fashion must receive policies and associate outputs of those policies.

If a model has a large run time (approximately 30–60 seconds) for a single iteration, typical analysis techniques will be quite computationally expensive. Simulation optimisation, evolutionary optimisation, or design of experiments become computationally expensive as the model becomes large.

Factor screening is a popular method for determining important inputs based on designed experiments and statistical analysis of those experiments. This is done in order to search through all model inputs i for an important subset of factors k, where $|k| << |i|$ (Bettonvil and Kleijnen, 1997; Montgomery, 1979; Tauer, 2009). In most literature, these subsets are predetermined by some analysis of the main effects of inputs (Bettonvil and Kleijnen, 1997; Mauro and Smith, 1984). Kleijnen (1975) provides a good overview of these screening techniques. Due to these computational requirements, it is not always possible to perform a large-scale experimental design in a preprocessed fashion. For example, if a large-scale simulation model has 10,000 controllable inputs. If that model has a single run average time of 60 seconds, a standard full factorial DOE of $2^{10,000}$ runs of the model would take approximately $4.99 * 10^{3,010}$ days to complete. A half factorial, thus eliminating half of the controllable inputs from consideration would take $3.53 * 10^{1,505}$ days to complete. For a much smaller example, consider a model with 1,000 controllable inputs. Using this model, a standard full factorial DOE would require $1.07 * 10^{301}$ runs and take $7.44 * 10^{297}$ days to complete. These methods of DOE would simply take too long to complete a factorial or fractional factorial on a simulation with as long of a runtime and as many input factors as some of the large simulation, models that certain domains may utilise.

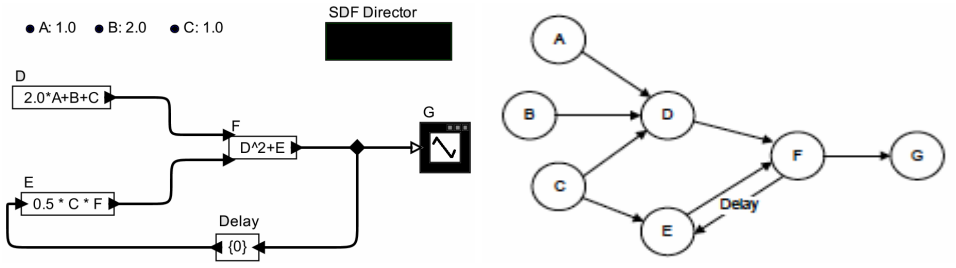
The last step in a MCDM process is the visualisation of results. Korhonen and Wallenius (2008) provide an overview as to the past work in the visualisation of MCDM results. In their overview, they discuss the use of scatter plots, bar charts, as well as radar/spider charts. Simple and recognisable charting methods such as a scatter plot or response surface work when visualising MCDM results with only two or three dimensions. Unfortunately, beyond three dimensions, standard charting methods fail. Spider charts allow the DM to view the impact of alternatives on several outputs at once.

Andrews (1972) curves and Cherno (1973) faces are two techniques that have been used to visualise multi-dimensional data. Agrawal (2005) and Chiu (2009) developed a hyperspace diagonal counting method and hyper-radial visualisation approaches for visualising many dimensional data. While the previously mentioned work in several fields gives aid to presenting multi-dimensional data, a main work in MCDM, that includes the DMs preference information comes from Korhonen (1988). Their Pareto race work allows a DM a visual method for exploring the Pareto frontier. Korhonen et al. (1997) also performed some work on a quadratic Pareto race. These methods and their approach to interaction with a DM in order to allow that DM the ability to manoeuvre around the Pareto frontier provide us motivation and techniques to use for our visualisation approach for MCDM results. When producing many potential solutions, it is best to allow the DM the ability to traverse those solutions in a manner that will allow them to select the best solution for them. It should be noted that in our PSO problem, the DM may choose to select a dominated policy if it satisfies some additional interests. If a dominated solution has minimal change from the baseline values, then it might be of interest to the DM to select a policy that is easy to obtain, as opposed to spending too much effort to reach a solution that is slightly better, but may be very difficult to reach. In our framework, we allow this decision to take place.

3 PSO framework

The PSO problem is the process of selecting a small subset of inputs of a simulation model to modify from their default settings in order to have a desired effect on a set of outputs of interest to a DM. Figure 2 shows the overall framework for our PSO solution methodology. The first step in the procedure is to elicit the underlying graph structure of the simulation model. The models that are utilised in this research are synchronous data flow (SDF) simulation models. A small sample SDF model and the associated elicited graph can be seen in Figure 1. SDF simulation models differ from discrete event simulation models in the scheduling of execution. Discrete event simulation has modules that are executed on a predetermined global schedule (Lee and Sangiovanni-Vincentelli, 1998). SDF modules execute as a reaction to the existence of data as inputs (Lee and Messerschmitt, 1987). SDF tools are used in many areas in the social sciences or engineering. Some specific examples are shown by Goldberg (1958) to be control theory problems, econometric models, queuing problems, and behaviour learning. The third box in the pre-processed row of the framework is experimental sampling. This sampling varies based on which approach is used. The sampling associated with the MDA is seen in Section 4. The sampling associated with the MSA is seen in Section 5. These sampling procedures are implemented to determine how changes in the inputs, whether it is at the node level in model decomposition or model level in model sampling, effect the outputs. Preprocessed sampling is defined as any sampling procedure that can be run on an SDF model. At this point, the DM will select their outputs of interest for a specific problem. This information is provided to the mathematical model generator. This model is solved with a single set of weights that can be provided by the DM (a single point estimate of the efficient frontier), or it generates an estimation of the entire efficient frontier using multiple sets of weights. These solutions will aid the DM in making policy decisions.

Figure 1 Sample SDF model and elicited graph, (a) sample SDF model (b) elicited graph



(a) (b)

Source: Tauer et al. (2010)

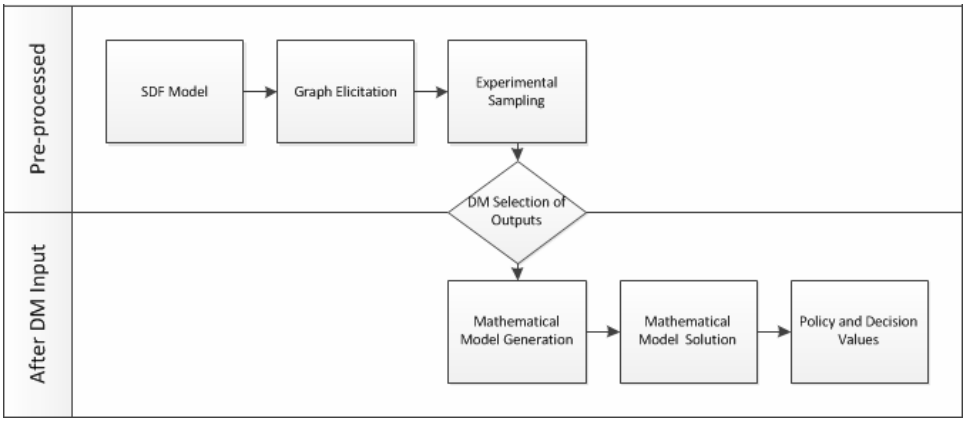
Our solution methodology for the PSO problem aims to minimise the computation time required from the point at which a DM selects the outcomes of interest, to when they receive solution policies to choose from.

This research takes two approaches for solving this multi-criteria PSO problem. The first approach is called the MDA. This approach requires the decomposition of the large simulation model into smaller black-box simulation models. These smaller models (nodes) are sampled and that sampling data is regressed upon to form constraints in a

MDA mathematical model. This nonlinear program is solved via the penalty-successive linear programming (PSLP) method and the solution to this is a solution policy.

The second approach is the MSA. This approach samples the entire model simultaneously and the data that is generated creates the MSA mathematical model.

Figure 2 Policy set optimisation framework



The benefit of this integer program is that it provides a method to limit the number of inputs in a solution policy. The resulting solution can then be used as an input to a local-search method. Solution time reduction techniques are examined for this mathematical model.

Both approaches are then used on a small sample simulation model and analysis of these results show that the ability of the MSA to limit the number of inputs is integral for selecting usable policies for a DM. Visualisation of all validated policies is discussed and a description of how this visualisation approach aids the DM in selecting their preferred policy is provided.

4 Model decomposition approach

The first method for selecting a policy for a large-scale SDF simulation model is using the MDA. A policy is defined as a set of inputs that should be changed from their default values in order to have some desired effect on certain outputs of interest.

Since we are working under the assumption that each SDF simulation model is a black-box simulation model, the MDA is based on the decomposition of the overall model into separate black-box functions. The breaking down of the model is completed by leveraging the graph structure that exists in an SDF simulation model. Exploitation of this graph structure does not impact the proprietary nature of these large-scale black-box simulation models. The actual computations performed in these black-boxes are not compromised in this procedure. The process is done by leveraging the work in Tauer et al. (2010). By doing this, we allow for estimations of each decomposed black-box function separately.

Initially, the model is exercised as a whole using a random balance experimental design (Satterwaite, 1959). Our random balance experimental design is an experiment that exercises the simulation a specific number of replications to determine potential ranges on each internal edge of the SDF graph structure. These random balance simulation runs allow for each input in the model to be randomly assigned (50% chance) to the associated low or high value. When these replications finish, each edge in the elicited graph has a recorded range. These ranges are used in the node sampling procedure.

Figure 3 shows an example of a node in a SDF model that has two inputs and how it would be sampled. These inputs can either be inputs to the entire model, or internal values to the overall model that are computed from a previous node. This node will have some sort of experiment exercised on it. The values from this experiment will then be regressed upon. If the experiment is a full factorial design, it is still much faster than performing a large-scale factorial design on the entire simulation model, due to only a single actor being exercised. With this speed increase, it allows us to perform any type of experiment we may want to perform. This could include a 2^n experiment where n is the number of inputs to the node, or perhaps even a higher order experiment such as 3^n or even 5^n. The example data table for the example node sampling in Figure 3 using a standard 2^n full factorial experiment can be seen in Table 1.

Figure 3 Node sampling

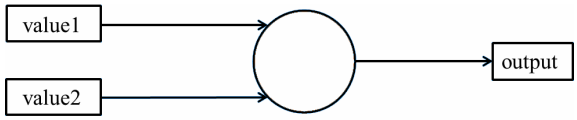


Table 1 Node sampling data table

Value1	*Value2*	*Output*
High	High	
High	Low	
Low	High	
Low	Low	

At this point in the process, every node in the SDF model will have data similar to the data in Table 1. The data in Table 1 then undergoes the regression procedure. The steps of this regression procedure are as follows:

1 linear regression

2 separable quadratic regression

3 quadratic with pairwise interaction regression

The first step is to process every remaining node using linear regression. This linear regression is the standard linear regression with each input to the node representing one term in the regression equation. Equation (1) shows what the sample linear regression equation would look like for the node in Figure 3.

$$output = \beta_0 + \beta_1 * value1 + \beta_2 * value2 \tag{1}$$

With any node that has an R^2 value below the set threshold, it is then passed to regression using separable quadratic terms. Equation (2) shows what the sample separable quadratic regression equation would look like for the same sample node.

$$output = \beta_0 + \beta_1 * value1 + \beta_2 * value2 + \beta_3 * value1^2 + \beta_4 * value2^2 \tag{2}$$

Any node that once again has not passed the set R^2 threshold, is processed using pairwise quadratic terms. Equation (3) shows what the sample quadratic regression with pairwise interaction would look like for the sample node.

$$\begin{aligned} output = {} & \beta_0 + \beta_1 * value1 + \beta_2 * value2 + \beta_3 * value1^2 \\ & + \beta_4 * value2^2 + \beta_5 * value1 * value2 \end{aligned} \tag{3}$$

If any node still does not meet the threshold, it can either continue onto further levels of adding terms into the regression (pairwise divisible terms, etc.) or the DM can set a stopping point for adding terms. Our stopping point is currently set to this pairwise quadratic level of regression. The reasoning for this stopping point is that in our sample model, a threshold of 0.975 had every node stopped at these three levels of regression. It is obvious that in other models, this threshold and these levels might not be sufficient. One of the advantages of the MDA is that the threshold and these types of regression terms is fully customisable and can be altered based on the type of model that is being analysed.

After every node has a set regression equation as an estimation, the MDA mathematical model is developed based on these equations. The purpose of the MDA mathematical model is to identify the set of inputs that most effects the outputs of the simulation model. Table 2 shows the parameters, sets and variables defined for the MDA mathematical model. The MDA mathematical model formulation is then shown.

Table 2 MDA parameter, set and variable definitions

w_g	Objective weight associated with output g
I_g	1 if output g is to be maximised, –1 if it is to be minimised
x_{it}	Value of input, function, or output i for time unit t
z_g^*	Goal value for output g
G	Set of outputs of interest in the elicited graph
L	Set of black-box functions defined by linear estimations
SQ	Set of functions defined by separable quadratic estimations
QPI	Set of functions defined by quadratic pairwise interaction estimations
I_{select}	Set of all inputs to the model that are allowed to be in a policy solution
(l_i, u_i)	Lower and upper bounds for $i \in I_{select}$
β	Regression estimation coefficients

$$\begin{aligned} &\min \ \alpha \\ &\text{subject to}: \alpha \geq \omega_g I_g \left(z_g^* - x_{g2} \right) \end{aligned} \qquad \forall g \in G \tag{4}$$

$$\beta_{0j} + \sum_{i \to j} \beta_{ij} x_{i1} = x_{j1} \qquad \forall j \in L \tag{5}$$

$$\beta_{0j} + \sum_{i \to j} \beta_{ij} x_{i2} = x_{j2} \qquad \forall j \in L \tag{6}$$

$$\beta_{0j} + \sum_{i \to j} \beta_{ij} x_{i1} + \sum_{i \to j} \beta_{i^2 j} x_{i1}^2 = x_{j1} \qquad \forall j \in SQ \tag{7}$$

$$\beta_{0j} + \sum_{i \to j} \beta_{ij} x_{i2} + \sum_{i \to j} \beta_{i^2 j} x_{i2}^2 = x_{j2} \qquad \forall j \in SQ \tag{8}$$

$$\beta_{0j} + \sum_{i \to j} \beta_{ij} x_{i1} + \sum_{i \to j} \beta_{i^2 j} x_{i1}^2 + \sum_{i \to j} \sum_{h \neq i | h \to j} \beta_{[hi]j} x_{h1} x_{i1} = x_{j1} \qquad \forall j \in QPI \tag{9}$$

$$\beta_{0j} + \sum_{i \to j} \beta_{ij} x_{i2} + \sum_{i \to j} \beta_{i^2 j} x_{i2}^2 + \sum_{i \to j} \sum_{h \neq i | h \to j} \beta_{[hi]j} x_{h2} x_{i2} = x_{j2} \qquad \forall j \in QPI \tag{10}$$

$$l_i \leq x_{i1} \leq u_i \qquad \forall i \in I_{select} \tag{11}$$

$$\begin{aligned} &x_{i1} = x_{i2} && \forall i \in I_{select} \\ &\omega_g > 0 && \forall g \in G \\ &\sum_{g \in G} \omega_g = 1 \end{aligned} \tag{12}$$

Our MDA mathematical model is a two-time-step approach to approximating the simulation model's inputs effects on the outputs. Originally, our approach to the mathematical model was to estimate the effects of each node at every time step with a higher level of fidelity. This lead to a very large mathematical model with a level of fidelity that was not required for our overall purpose of the mathematical model.

In the mathematical model formulation, minimising α and constraint (4) minimises the maximum distance from the goal values for each output of interest g. Constraints (5) and (6) are associated with the regression estimations of black-box functions, for time period one and time period two, that fall within the set of nodes L that are estimated using linear regression equations. Constraints (7) and (8) are for the set of nodes that fall within the set SQ that are estimated by the separable quadratic regression equations. Constraints (9) and (10) are for the set of nodes QPI that are estimated by the quadratic regression equations that include pairwise interaction. All six of these constraint sets estimate the output of a node based on the inputs provided to it. Whether those inputs are inputs to the entire model, or outputs from other nodes in the SDF model, these constraints work the same way.

Constraint (11) ensures that the policy inputs (x_i where $i \in I_{select}$) are chosen to be at values that are within their allowable range. Constraint (12) forces those policy inputs to then remain at that chosen value for the second time period. All other x_i where $i \notin I_{select}$ are allowed to move in any direction at any magnitude.

Since this mathematical model is nonlinear due to constraints (7), (8), (9) and (10), a nonlinear solution technique must be used. Since this problem has a highest order of two, we have chosen to use the penalty successive linear programming (PSLP) algorithm (Bazaraa et al., 2006).

The PSLP algorithm is based on starting at some initial estimate of the optimal solution. Our initial estimate is the solution of the original problem replacing the nonlinear regression equations with their linear regression equations.

After the values are initialised, the method then solves a sequence of first-order approximations of the nonlinear model. Since these are the linear approximations of the nonlinear model, they are solved very efficiently. Since there is no requirement for the remaining linear constraints to bound the problem, trust regions must be used to ensure convergence of the algorithm (Bazaraa et al., 2006).

This research utilises this solution method for the nonlinear program because of the main advantage it provides. Its ease and robustness for implementation in large-scale problems is the first advantage. Some large-scale simulation models produce very large mathematical models and this advantage is key in the implementation of the solution method for these models.

PSLPs disadvantages that often hinder its use also do not impact our implementation. SLP algorithms often see slow convergence to the optimal point when it does not exist on a vertex. Also, SLP algorithms typically violate the nonlinear constraints en route to optimality. While these two disadvantages may be of concern for most implementations, it is not of concern to this problem. Our mathematical model is simply an estimation of a simulation model that is an estimation of reality. These multiple levels of estimation allow for some leeway in this aspect.

The solution of the PSLP algorithm with a single vector of weights, ω, provides a single point estimate of the efficient frontier. This may be of interest to a DM if they have a set of weights already in mind for the problem. Varying this vector can provide a different estimation of the frontier.

The MDA within the PSO framework allows for the DM to interact with the system and change certain parameters. The DM is responsible for setting the indicator vector I, identifying whether an output of interest g is maximised or minimised. The DM is also responsible for selecting the set of these outputs, G.

5 Model sampling approach

Our next approach to solving the PSO problem is the MSA. The MSA is centred around a change in the sampling technique that is used to develop the model data. The MDA as it currently exists does not limit the number of inputs selected in the policy solution. This can be done heuristically, but it is of interest to have a solution method that handles the limitations of the policy size automatically. Another advantage of the MSA is that it does not require decomposition of the simulation model.

The purpose of this approach is to provide policies that are more realistic than what is currently produced by the MDA. This means that the MSA mathematical model can produce policies for a simulation model that would be more feasible to implement and of interest to a DM. It is obvious that a policy consisting of a large number of modified inputs is not feasible for a complex system, and certainly not in a real application.

Another feature of the MSA is that it can be used to create many policies. Generating many alternative policies would be of interest if one were to leverage parallel computing and local search to intelligently search the input space of the simulation model. Having multiple policies is also of interest when visualising the results and allowing the DM to choose their preferred policy.

Sampling of the model as a whole is done with as few simulation runs as possible. This is due to the fact that the evaluation time of these complex SDF simulations can be close to or even well above 90 seconds per replication. This procedure requires a number of simulation runs equal to two times the number of controllable inputs in the simulation plus one. Assuming a model with a 60 second run time and 10,000 controllable inputs. This sampling procedure would take only 13.89 days as opposed to the $4.99 * 10^{3,010}$ days using a full factorial design or $1.07 * 10^{301}$ days when doing a $\frac{1}{10}$ factorial. It is worth noting that if the model is $\frac{1}{10}$ the size of the original model, it is full factorial would still take $1.07 * 10^{301}$ days. While this 13.89 days seems like a long time, it can be run (as completed on our real Department of Defense (DoD) application) on a parallelised setup and use several machines in a pre-processed fashion. A pre-processed simulation run is a simulation run that is performed before a DM has to interact with the system. Even if someone had a massive computer infrastructure, the runs required for a full factorial (or even a half factorial) on one of these large simulation models would not be feasible.

This sampling procedure equates to running the simulation once while all inputs are located at their default value (baseline run), and two runs for each input, one when it is assigned to the lower bound value (while all other inputs are held at their default values) and one when it is assigned to the upper bound value (while all other inputs are held at their default values). Each simulation run is completed while recording the value of every output in the model. This separable and linear assumption for the effect of input *i* on output *j* will be analysed when the MSA results are compared to the results of the MDA. Each input/output pair has a value triplet (VT) where $VT_{ij} = \{v_{ij}^L, v_j^D, v_{ij}^H\}$ where v_{ij}^S represents the value produced for output *j* when input *i* is at setting *S*. The options for settings are low (*L*), default (D), and high (*H*). Notice that for the default setting, this is independent of input *i*. This is because for all *i*, v_{ij}^D is the same. All inputs *i* are at their default, thus being independent of *i*.

The purpose of the MSA mathematical model is to select a small subset of inputs to modify and the direction in which to modify them in order to have the largest estimated impact on the desired outputs. This model utilises the Tchebycheff method, which prevents the exclusion of non-dominated solutions that might be missed using the regular weighted objectives method. These solutions correspond to convex dominated solutions.

Table 3 defines the parameters and variables used in the MSA mathematical model.

Table 3 MSA parameter and variable definitions

w_j	Objective weight associated with output j
I_j	1 if output j is to be maximised, 0 if it is to be minimised
z_j	Mathematical model calculated value of output j
z_j^*	Goal value for output j
y_i	1 if input i is chosen to move from it is default value, 0 otherwise
y_i^L	1 if input i is chosen to be at the low value, 0 otherwise
y_i^H	1 if input i is chosen to be at the high value, 0 otherwise
c_{ij}^L	Change in output j caused by moving input i to the low value
c_{ij}^H	Change in output j caused by moving input i to the high value
N	The maximum number of inputs allowed to be in a policy solution

The formulation for the mixed integer program (MIP) that is generated in the MSA is seen in Table 3.

$$\begin{aligned} &\min \ \alpha \\ &\text{subject to}: \alpha \geq \omega_j I_j \left(z_j^* - z_j \right) && \forall j \in J \end{aligned} \tag{13}$$

$$v_{ij}^L y_i^L + v_j^D \left(1 - y_i^L\right) = v_j^D + c_{ij}^L \qquad \forall i \in I,\ j \in J \tag{14}$$

$$v_{ij}^H y_i^H + v_j^D \left(1 - y_i^H\right) = v_j^D + c_{ij}^H \qquad \forall i \in I,\ j \in J \tag{15}$$

$$v_j^D + \sum_i c_{ij}^L + \sum_i c_{ij}^H = z_j \qquad \forall j \in J \tag{16}$$

$$y_i^L + y_i^H = y_i \qquad \forall i \in I \tag{17}$$

$$\begin{aligned} &\sum_i y_i \leq N \\ &\sum_j \omega_j = 1 \\ &\omega_j > 0 && \forall j \in J \\ &y_i^L, y_i^H, y_i \in \{1, 0\} && \forall i \in I \\ &c_{ij}^L, c_{ij}^H, z_j \text{ are unrestricted} \end{aligned} \tag{18}$$

The objective function of α is minimised, and constraint (13) finds the weighted deviation from each output j's goal. The combination of these two is the Tchebycheff minimisation of the maximum distance from some goal z^*. Constraints (14) and (15) estimate the change in output j caused by moving input i either to the low value or high value respectively. The summation of these changes across all inputs calculates the

estimated output value z_j for all outputs j in constraint (16). The restrictions on the use of inputs are in Constraints (17) and (18). Constraint (17) restricts that an input i (if selected in a solution policy) can only be and must be moved in one direction (low or high). Constraint (18) restricts the number of inputs allowed to change from their default value according to a DM.

A solution of this mathematical model with a single vector of weights, ω, provides a single point estimate of the efficient frontier. Varying this vector of weights provides a varied estimation of the frontier. The spacing of these weight configurations will either increase the fidelity of the frontier estimation as well as the solution time by using more point-estimates, or can decrease both by using fewer point-estimates.

There are several parameters within the PSO system that a DM can change in order to effect the solution(s). The first parameter is ω_j. This can be altered as previously mentioned, by the selection of a single weight vector ω for a single point estimate of the efficient frontier, or it can be spread out at a DM's chosen spread to estimate the entire efficient frontier. Another parameter is the indicator vector I_j which identifies whether an output j is maximised or minimised. The DM sets this parameter while selecting their outputs of interest for the generation of the mathematical model. N represents the maximum number of inputs that can be chosen in a solution policy. It can be very difficult for a DM to handle the tradeoffs involved with a large number of inputs changing in any policy. For this reason, we have almost entirely chosen policy sizes of less than ten for our example problems.

Another useful aspect of the MSA mathematical model is that it can easily generate additional policies to create a database of policies. An input i can be included ($y_i = 1$) or excluded ($y_i = 0$) from a policy via adding one single constraint to the mathematical model. It can also generate additional policies by using recursive solution of a mathematical model with addition of a single constraint $\sum_{i \in SP} y_i \leq |SP| - 1$ where SP represents the previous solution policy. Addition of this constraint forces the model to select a new solution policy with each iteration of the problem.

Two approaches were taken to try and improve solution time for the MSA model. The first was input restriction and the second was policy size reduction. Table 4 shows the idea behind the two techniques for solution time improvement. Consider a scenario in which a simulation model exists and a DM is considering analysing this system using our PSO solution procedure. If the problem is solved using all possible inputs and a maximum policy size of seven, there is reason to consider solving the problem with fewer inputs and a smaller policy size to determine not only if the same frontier is found, but also if the solution time for developing that policy is reduced. In Table 4, each cell represents one combination of mathematical model manipulations to determine if reducing the number of inputs allowed in the mathematical model and reducing the maximum policy size allows for the system to still produce a good estimation of the efficient frontier and produce it in a smaller solution time. Moving down a row represents the reduction of the number of inputs allowed in the model and moving across a row is the reduction of the maximum policy size. The purpose of this section is to determine in this hypothetical scenario how far to the right and down in this box of cells we could consider moving our system parameter values and still maintain a good estimation of the efficient frontier.

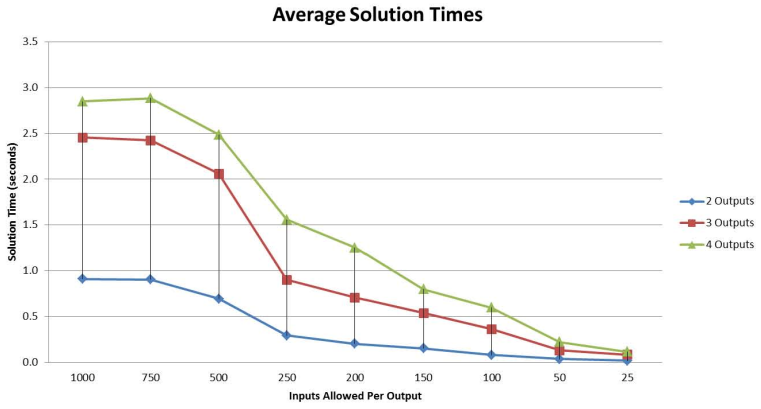
Table 4 Analyse effects of N and allowable inputs

7	6	…	n
All inputs	…	…	…
7	6	…	n
∪ Top 200s	…	…	…
7	6	…	n
…	…	…	…
7	6	…	n
∪ Top I*s	…	…	…

In order to compare how these methods effect the frontier estimation, we must compare the estimations created by each method and determine the closeness of the points. From this comparison, it was determined that at certain levels, the input restriction method greatly reduces the solution time while having minimal if any impact on the efficient frontier. Unfortunately, the policy size reduction method had a direct impact on the efficient frontier and should be implemented with consideration of that. Additional detail on these methods can be seen in Myers (2013).

Since the MSA does not require use of a proprietary simulation model for decomposition, it is possible to show computational results with larger test data. The test scenarios used were data that contained 1,000 controllable inputs. Figure 4 shows the average solution times in these test scenarios. The average solution time without the use of the input reduction technique is under 3 seconds in all three scenarios. With the inclusion of the input reduction technique, that solution time drops well below 1 second in all scenarios.

Figure 4 Input reduction solution time improvement (see online version for colours)



It is important to determine if the use of the input reduction technique affects the estimation of the Pareto frontier. Figure 5 shows the frontier estimations based on each of the input reduction values for scenario one. It is easy to see that the frontier estimation for this test scenario is very good with all values.

Figure 5 Input reduction frontier estimation: scenario (see online version for colours)

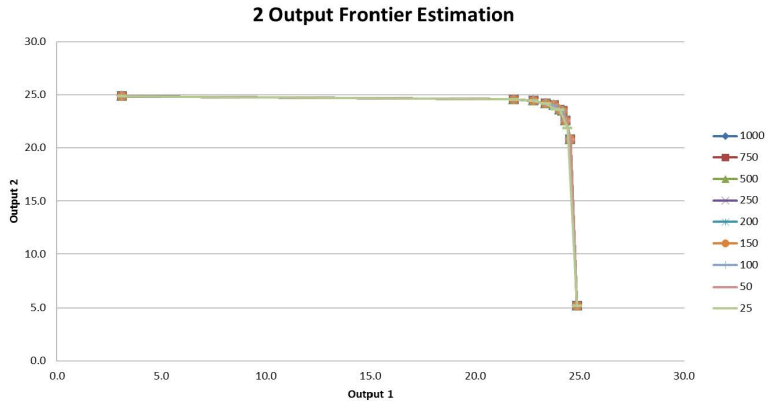
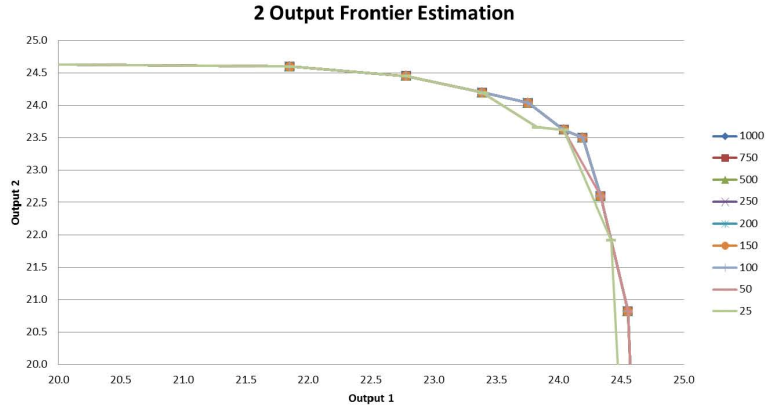


Figure 6 shows a close-up of the central portion of the frontiers to show that at the lowest values of the changed variable, the frontier estimation does in fact change.

Figure 6 Input reduction frontier estimation close-up: scenario 1 (see online version for colours)



The importance of this information is that the PSO process, when utilising the MSA, provides an estimation of the Pareto frontier to the DM in a very short amount of time. This allows the DM to analyse and interact with the policies in order to determine the preferred solution policy.

6 PSO system

In order for a DM to interact with the PSO solution system, it is assumed that all pre-processing is completed on the model. This means that all of the model's sampling data is accessible. The first step for the DM is to select whether to utilise the MDA or the MSA for policy development. The DMs next step is to select the outputs of interest. After these outputs of interest are selected, the DM must set the values for the indicator

function. This means that they must select whether each output of interest is to be minimised or maximised by the solution approach being used.

The DM then must decide which piece of data they are interested in using for making their decision. The two options provided in our test scenarios was an individual data point at the end of the simulation run (last time interval) and area under the curve. We only provided these two options because we believed it accurately represented how a DM would be interested in making a decision in our test cases, but also represents that any data type (individual point, area under curve, average across a time interval, etc.) could be utilised.

The DM must select whether they are interested in generating a policy for a single set of weights for the outputs of interest or if they are interested in estimating the entire efficient frontier using multiple weight configurations. If they select an individual set of weights, they must provide that set of weights for the mathematical model. If they select the estimation of the entire frontier, they must select the number of weight configurations for each output (i.e., four weight configurations per output would generate weights of ϵ, 0.3333, 0.6666, and 1.0 for each output).

After the DM interaction with the system is completed, the mathematical model must be solved using the weighted objectives formulation (and not the Tchebycheff approach) with the 'end-point' weight configurations. An 'end-point' weight configuration is a weight configuration where the only terms are one or ϵ. This allows the system to develop the unattainable goals that are required by the Tchebycheff formulation. Using this method, the goals may be individually attainable, but collectively unattainable. We use a factor of 1.1 to guarantee independently unattainable goals and guaranteed collectively unattainable goals for our system.

If the goals were generated by the DM, it raises the chances that the goals would be achievable. An achievable goal does not allow the math model to get the best possible policy, just the one that gets closes to the goal set by the DM. While this may be of interest to some DMs, we feel as though it would better suit a DM to have these goals be unattainable.

The solution to each Tchebycheff problem (whether it is a single weight configuration or full efficient frontier estimation) is a solution policy. These policy then needs to be validated in order for the actual impact of that policy on the outputs of interest can be identified.

A validated policy is defined in this system as a policy that has been evaluated by the simulation model. Evaluation by the simulation model allows a DM to understand how the system will react to a given solution policy. In order to validate one of the solution policies that are generated by a mathematical model, we must exercise the simulation model once with the given policy. Effects of the new policy on the system can be evaluated by recording the output values that are produced when performing this simulation run.

The preferred form of policy validation would be to include it as a starting point for a local search method. If the policy is developed using the MDA then the number of inputs included in the policy is too great for local search. This large policy size would require many simulation runs for the local search method. If the policy is developed using the MSA then the Nelder-Mead (NM) local search method is used for validation of policies. This also generates new candidate solutions for the DMs consideration.

A NM search procedure is performed after the selected inputs are provided by the mathematical model. It is shown in the literature that in a linear or convex nonlinear system, NM local search will find the efficient frontier (Barton and Ivey, 1996; Nelder and Mead, 1965; Spendley et al., 1962). It has also been modified and used in the literature for simulation optimisation purposes in its modified form (Kuriger and Grant, 2011; Kuriger and Ravindran, 2005). In Kuriger and Grant (2011), a Lexicographic Nelder-Mead simulation optimisation (LNM-SO) approach is taken for the multiobjective simulation optimisation problem. Unfortunately, the LNM-SO approach requires DM interaction as well as a lot of exercising of the simulation model in order to generate an estimate of the efficient frontier.

7 Policy visualisation

Visualising results is a crucial aspect specifically for the second part of any MCDM process, aiding the DM in selecting their most preferred solution among the set of possible solutions.

Both mathematical model approaches produce solution policies. These solution policies can be stored in a database of potential policies from which the DM can select their most preferred. Assuming that there exists a large database of validated policies available for a specific model, it is important to provide a way for a DM to interact with all of these potential policies and aid them in not only selecting the most attractive policy for them, but also to explore other potential policy options.

Figure 7 MCDM results visualisation approach

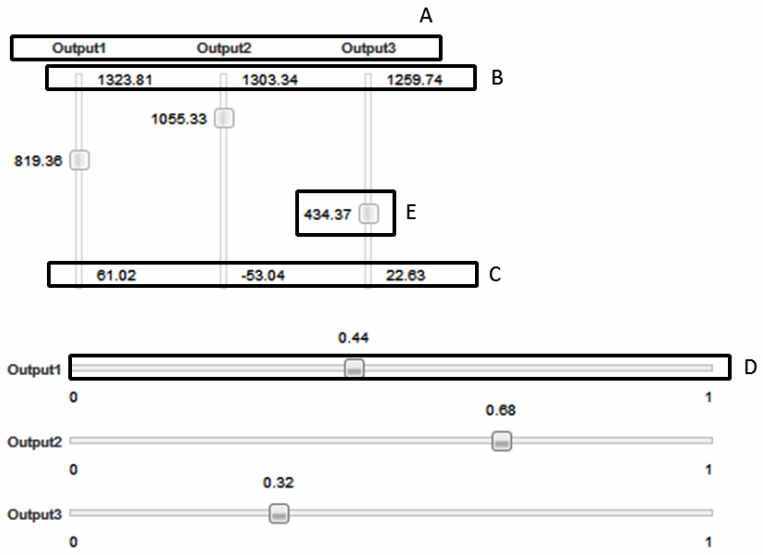


Figure 7 shows an example of what our visualisation approach is for the MCDM results from the MSA and MDA. Box A in Figure 7 is where the output names are located. Our visualisation approach uses vertical slider bars. One slider bar exists for each output of interest selected by the DM during the PSO process. Box B is the 'best' value for each

output that is found in the database of validated policies. This value is the maximum value if the output of interest is supposed to be maximised and the minimum value for each output that is supposed to be minimised. These values are determined with a simple search of the policy database. Box C is the 'worst' value for each output from the database. This 'worst' value corresponds to the exact opposite value as what is determined to be in box A. This is to make it easier for the DM to visualise the improvement or deterioration of an output value as different policies are explored.

Our visualisation approach also uses horizontal slider bars for the weights associated with each output of interest. Similar to the vertical sliders, there is a single slider bar for each output of interest. Box D is a single weight slider associated to a single output of interest. The DM would slide these weight sliders into values that they think represent their preference trade-offs for the outputs. The system would then normalise these weights to one and search the database for the 'optimal' policy for those weights. This policy is selected based on using the Tchebycheff evaluation mentioned in the previous sections of this paper. Box E is the value from the database for the resulting optimal policy. Since we are simply applying a weighted objective function to each policy in the database, this computation is very fast, allowing for almost real time sliding of the weight bars with nearly instantaneous results.

One consideration is any 'ties' produced in the output evaluation. If a tie were to exist, there must be a method that allows the DM to view and potentially select each of the policies that are tied with the presented policy.

This initial approach, using DM selected weights in order to search the database, only shows non-dominated policies. Non-dominated policies, while mathematically preferred, may not be the most preferred solution for a DM. For example, one policy of interest may be the policy with minimal changes from the baseline of the original system. This visualisation approach can easily be extended to allow DM to further search the database of potential policies for a most preferred suboptimal solution policy.

Additional features that could be extended from this visualisation approach to aid the DM in selecting their most preferred policy include the presentation of the policy values near the results shown. Since a realistic policy created by the MSA has a small number of changed inputs, this could be completed by just listing the changed inputs.

The next step after the presentation of the policy values would be to allow for changes to those policies. This can be done in any capacity and once a new policy is established, there are several options. The first option would be to search the database for the 'closest' existing policy. This can be done with any measure of closeness that is desired.

Another option is used if the chosen policy is not a non-dominated policy. This option is to find any of the existing policies that dominate the newly chosen policy. It is likely that if a DM is selecting policies by hand, they will likely create dominated policies. It would be a simple evaluation of the existing database to determine which of the database policies dominates the new policy. If there are multiple policies that dominate the new policy, the method for showing tied policies would be used.

The next extension is to simulate this newly developed policy. This would be of interest to determine the exact system reaction to the changes the DM is interested in making. One advantage of this approach is that it allows the DM to add simulation runs to the database that they are interested in. While these runs may not end up as non-dominated, if they are selected by the DM, they are obviously potentially a preferred solution.

There are other features that would allow the DM to control what policies are shown. These features would also not guarantee a non-dominated solution for the overall PSO problem. The first feature would be the inclusion of forced minimum or maximum allowable threshold bars. There would be one bar for minimum and one bar for maximum for each output of interest. These bars could be set at any level and force any shown policy to meet the requirements. One possible result is that there could be no policies in the database that meet the requirements set by the DM.

The DM could also select to use % deviation boxes around the output values of the optimal policy. This would then search the database and determine if there are any policies that exist that produce output values within the % deviation boxes. This is of interest if the DM would like to see similar output values, but dislike the policy that creates those output values. Similarly to the above method using minimum and maximum bars, there could be no policies in the database (other than the currently shown policy) that meet the requirements.

The visualisation method shown in Figure 7 allows not only for the DM to visualise the changes in the output values in many dimensions, but could also allow the DM to modify the policy that creates the shown output values. This ability to modify the policy, create new policies in the database, and visualise the policy outputs aid the DM in selecting their truly preferred policy for the analysed system.

8 Results and conclusions

The difficulties associated with the MDA for solving the PSO problem are mentioned throughout this paper. The requirement for model decomposition and the inability to limit the number of inputs allowed in a solution policy without making it a nonlinear integer programming problem caused difficulties for testing this approach. Models that are large enough to test the limits of the MDA are proprietary and thus the results cannot be published in this paper. Therefore the analysis of the MSA shown below are using test data from a large-scale model, and the method comparison is shown using a small sample test model that does not truly test the limits of either approach in terms of solution capability.

The simulation model that is used as our sample model is based on a simulation model that is similar to the example model used in Tauer et al. (2010). This model is a version of the Lotka-Volterra system of multiple species and their interactions. It is based on the concepts located in Wangersky (1978) and Harrison (1979). Here, the model will be referred to as the population model. This model is similar to these models, but it is only based on the combination of concepts from both to build a larger simulation model.

As previously stated, the target application of this work is a large-scale simulation model that contains many controllable inputs. While results from testing on actual large-scale models of this size are proprietary, we present a sample model and results associated with that sample model to show the effectiveness of finding a set of good policies for a DMs consideration. It is recognised that this is a relatively small simulation model to be performing these analysis on, but the results showing effectiveness of the solution techniques are shown. Our objective is to minimise computation time required to generate good policy sets as well as reasonable policy sets for a DMs consideration. A

reasonable policy set is a policy set that is implementable by a DM (not too many inputs to manoeuvre in order to utilise).

This version of the population model contains 20 populations. Ten of these populations are predatory in nature, while the remaining ten are considered prey for these predatory populations. The population of the prey populations grow at each time interval and do not die from natural causes. Each predatory population decreases at each time interval and they do not reproduce on their own. Predatory populations interact with prey populations by consuming them. When this occurs, the population of the prey decreases and the predator population increases. All predators prefer to eat a single associated prey, but will however consume any prey with varied levels of success. This leads to two controllable inputs for each prey population and seven controllable inputs for each predator population. Twenty total populations means that there are 90 controllable inputs in the system. Ten of these inputs are unconnected, to insure that inputs are not selected to be in a policy unless they actually benefit the system.

Since the MDA requires that the simulation model be decomposed, we have modelled this system using a SDF simulation approach. This type of simulation modelling lends itself to decomposition and sampling as needed in order to use the MDA. In total, there are 350 nodes in the decomposed simulation model.

Each population within the model is represented by a series of actors with computations to determine the effects on each population throughout the period of one iteration. Each iteration in the sample model represents one day of interaction between the populations. For our example problems, every simulation run is associated with 5000 days. This is equivalent to a run time of about ten seconds per iteration. Ten seconds per iteration and 90 controllable inputs would require 290 simulation runs for a full factorial experimental design. This would take approximately $1.43 * 10^{23}$ days to complete. So even for such a small simulation model, the full factorial design would take a very long time to complete.

We developed two example problems from this simulation model. The first example problem is the maximisation of two populations, one from each category. The second example problem is the maximisation of three populations, adding a second population from the prey category. These three populations were hand chosen because they corresponded to the most nonlinear nodes in the decomposition procedure.

All test cases and solutions for analysis were coded using Java and a Java wrapper of GLPK, version 4.47. They were solved on a computer using an Intel Xeon W3530 Quad-Core 2.8GHz CPU with 8 GB of RAM.

Since this simulation model is a model of an ecosystem, it is important for the system to remain stable. Stability in this system is defined as ensuring that all population values remain above zero during model sampling for both the MDA and MSA. To insure that this occurred, we set the model sampling percentage to 10%. This means that for the random balance simulation runs for the MDA node sampling and for the model sampling input manipulation, each input's high value was 5% above it is default value, and it is low value was 5% below it is default value. This proved to be the largest percentage of input movement that did not cause any population values to reach zero.

Example problem one was solved with 11 weight configurations. This leads to 11 solutions to consider. Example problem two was solved with four weight configurations. This leads to ten solutions to consider.

In the decomposition phase there were 350 nodes in the decomposed model. Ninety of these nodes are input nodes, which leaves 260 nodes to go through the regression procedure. In our sample model decomposition, we used a 0.975 threshold for the R^2 value. Using this threshold, 249 nodes were estimated by the linear regression equation, while 11 nodes went through the separable quadratic regression and into the quadratic with pairwise interaction regression equation. So in total, 90 nodes were input nodes, 249 nodes were linear, and 11 nodes were nonlinear.

The largest drawback of the MDA mathematical model is that it does not have a built-in method of limiting the number of inputs that it selects to be in the solution policy. Since our sample problems deal with a 20 population ecosystem, we added extra constraints to the mathematical model to insure that the mathematical model does not purposely eliminate any population from the ecosystem with the solution policies that it produces. This serves two purposes. First, to try and create solution policies that do not eliminate any populations in the ecosystem, but also to try and limit the number of inputs selected in a solution policy.

In the MDA mathematical model for the two example problems, there are two continuous variables and two constraints for each node in the decomposed simulation model. Therefore for the MDA mathematical model, there are 700 variables and 700 constraints. We also add the 20 constraints for the population values described above. Therefore, there are 720 constraints in the model with the extra 20 being the prevention of any population from reaching zero. After decomposition of these nodes, 11 of the 720 constraints are nonlinear.

In the MSA mathematical model, the number of variables and constraints are based on the number of controllable inputs in the system as well as the number of outputs of interest in the model. Table 5 shows the number of binary variables and constraints in the MSA mathematical model from each one of the sample problems.

Table 6 compares the solution times associated with each example problem for both the MDA and MSA mathematical models.

Table 7 shows the average number of inputs selected to be in a solution policy for each of the example problems. It is already known that the MDA does not consider the number of inputs that changed from their default values. It can be seen from Table 7 that the parameter N in the MSA mathematical model allows for the guarantee that a solution policy will be of a manageable size. When we tested the MDA model and left the extra 20 specialty constraints out of the formulation, the average policy size went up to 58.7 and 74.9 and created an unstable system.

Table 5 MSA mathematical model size: example problems

Example problem	*Inputs* $\|I\|$	*Outputs* $\|J\|$	*Binary vars.* $3\|I\|$	*Constraints* $\|I\| + 2\|J\| + 2\|I\|\|J\| + 2$
1	90	2	270	278
2	90	3	270	281

Table 6 Method comparison: solution times

Example problem	*Avg. MDA sol. time*	*Avg. MSA sol. time*
1	0.3054	0.4032
2	0.3278	0.4154

Table 7 Method comparison: solution policy sizes

Example problem	*Avg. MDA policy size*	*Avg. MSA policy size*
1	26.6	5
2	38.9	5

The most important result of the analysis performed on the sample model using both solution approaches is that the MDA created unstable policies. The unstable policies were a direct result of selecting too many inputs to modify in the system.

The MSA created solution policies in nearly as short amount of time and allowed the system to remain stable with all solution policies. While these policies were not as lucrative as the MDA policies for the system when only considering the outputs of interest, the large policy size created unacceptable unintended consequences.

After analysing the solution times and policy sizes associated with each of the solutions, the next consideration is the validation of policies. The only method of policy validation for the MDA is using a single run of the simulation model using the solution policy. There are two types of validation possible for the MSA. The first type is the single simulation run approach used by the MDA and the second is the use of the solution policy as a starting point for NM local search.

Tables 8 and 9 show the output values for the validated policies for each example problem. Table 8 shows the output values for the validated policies for the MDA, MSA, and NM local search using the MSA as a starting point. These values are for the first example problem. It can be seen in Table 8 that the values produced for the outputs of interest are greater for the MDA than for either of the other two methods. It is important to note however, that all of the policies created by the MDA created an unstable system and caused some unweighted populations to reach a value of zero. This is a result of the large number of inputs being altered in the solution policy. These unintended consequences of these policies make them unattractive to a DM.

Table 8 Method comparison: output values: problem one

Weight cofig.	*MDA*		*MSA*		*MSA → NM*	
	Predator2	*Prey6*	*Predator2*	*Prey6*	*Predator2*	*Prey6*
[1.0, ϵ]	1.5501	1.2455	1.3303	1.3014	1.3303	1.3014
[0.9, 0.1]	1.3211	1.9156	1.1945	1.9391	1.1965	1.9272
[0.8, 0.2]	1.3202	2.0155	1.1965	1.9272	1.1965	1.9272
[0.7, 0.3]	1.3105	2.098	1.1985	1.9497	1.1985	1.9497
[0.6, 0.4]	1.3308	2.1854	1.1855	1.9991	1.1857	1.9989
[0.5, 0.5]	1.3052	2.2986	1.1725	2.0451	1.1725	2.0451
[0.4, 0.6]	1.2877	2.4512	1.1723	2.0557	1.1725	2.0451
[0.3, 0.7]	1.2988	2.6277	1.1711	2.0199	1.1711	2.0199
[0.2, 0.8]	1.2744	2.7155	1.1704	2.1207	1.1704	2.1207
[0.1, 0.9]	1.1145	2.7122	1.0874	2.1332	1.0874	2.1332
[ϵ, 1.0]	1.0195	3.0199	1.0292	2.3666	1.0292	2.3666

Table 9 Method comparison: output values: problem two

Weight config.	*MDA*			*MSA*			*MSA → NM*		
	Predator2	*Prey6*	*Predator6*	*Predator2*	*Prey6*	*Predator6*	*Predator2*	*Prey6*	*Predator6*
[1.0, ϵ, ϵ]	1.5501	1.2455	2.0112	1.3303	1.3014	1.9878	1.3303	1.3014	1.9878
[0.66, 0.33, ϵ]	1.3308	2.1854	2.0785	1.1855	1.9991	1.9744	1.1857	1.9989	1.9754
[0.66, ϵ, 0.33]	1.3101	2.0977	2.0998	1.211	1.7852	2.2195	1.211	1.7885	1.2197
[0.33, 0.66, ϵ]	1.2988	2.6277	2.1145	1.1711	2.0199	1.9911	1.1711	2.0199	1.9911
[0.33, 0.33, 0.33]	1.2101	2.3211	2.1343	1.1845	1.9785	2.151	1.1845	1.9785	2.151
[0.33, ϵ, 0.66]	1.2465	2.1198	2.7845	1.1455	1.4512	2.4413	1.1644	1.4511	2.4429
[ϵ, 1.0, ϵ]	1.0195	3.0199	2.1197	1.0292	2.3666	2.0155	1.0292	2.3666	2.0544
[ϵ, 0.66, 0.33]	1.0355	2.6511	2.2356	1.0798	1.8712	2.5789	1.0795	1.8712	2.5799
[ϵ, 0.33, 0.66]	1.0498	2.4785	3.1113	1.1023	1.6545	2.4785	1.1025	1.6589	2.4788
[ϵ,ϵ, 1.0]	1.1412	1.7588	3.541	1.0301	1.34	2.7101	1.0301	1.34	2.7101

Table 9 shows the output values for the validated policies for the three methods on the second example problem. The same results can be seen in Table 9 as were discussed above for the first example problem in Table 8.

It is worth noting that in each of Tables 8 and 9 there are examples of an output value increasing as the weight on that output decreases. This is a result of the mathematical models not estimating the outcome of the simulation model properly. This was expected since the mathematical model is just used to generate policies for a DM to select their most preferred policy.

If the system were more nonlinear, however, the MDA can produce solution policies that are not at the extreme points of the input space. There are two approaches to solving this issue. The first approach is the use of local search using the MSA solution policy as the input. The results above show that for our two example problems, the NM search found the optimal policy to be very close to the extreme points used by the MSA.

The second approach would be to utilise the features of both approaches for a combined PSO solution framework. Assume a model was large enough and nonlinear enough to require the input selection abilities of the MSA, and the input value selection abilities of the MDA. In this model, it would be possible to use the MSA to select the inputs associated with the solution policy, and then to use the reduced MDA mathematical model to only allow those selected inputs values to change from their default in the overall, final solution policy.

Acknowledgements

This is approved for public release under Case Number: 88ABW-2015-3751.

References

Agrawal, G. (2005) *The Hyperspace Pareto Frontier for Intuitive Visualization of Multi-Objective Optimization Problems*, PhD thesis, University at Buffalo (SUNY), Buffalo, NY.

Aittokoski, T. and Miettinen, K. (2010) 'Efficient evolutionary approach to approximate the Pareto-optimal set in multiobjective optimization', *Optimization Methods & Software*, Vol. 25, No. 6, pp.841–858.

Andrews, D. (1972) 'Plots of high dimensional data', *Biometrics*, Vol. 28, No. 1, pp.125–136.

Barton, R. and Ivey, J. Jr. (1996) 'Nelder-Mead simplex modifications for simulation optimization', *Management Science*, Vol. 42, No. 7, pp.954–973.

Bazaraa, M.S., Sherali, H.D. and Shetty, C. (2006) *Nonlinear Programming: Theory and Algorithms*, 3rd ed., Wiley-Interscience, Hoboken, New Jersey.

Bettonvil, B. and Kleijnen, J.P.C. (1997) 'Searching for important factors in simulation models with many factors: sequential bifurcation', *European Journal of Operational Research*, January, Vol. 96, No. 1, pp.180–194.

Branke, J., Deb, K., Miettinen, K. and Slowinski, R. (2008) *Multiobjective Optimization*, Springer-Verlag Berlin Heidelberg, Germany.

Cherno, H. (1973) 'The use of faces to represent points in k-dimensional space graphically', *Journal of American Statistical Association*, Vol. 68, No. 342, pp.361–368.

Chiu, P-W. (2009) *The Hyper-Radial Visualization (HRV) Method for Visualization of the Hyperspace Pareto Frontier for Multi-Objective Optimization Problems*, PhD thesis, April, University at Buffalo (SUNY), Buffalo, NY.

Coello-Coello, C.A., Lamont, G.B. and Van Veldhuizen, D.A. (2006) *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, ISBN 0387332545, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Deb, K. (2001) *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley-Interscience Series in Systems and Optimization, ISBN: 9780471873396, John Wiley & Sons, West Sussex, England.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, April, Vol. 6, No. 2, pp.182–197.

Dyer, J., Fishburn, P., Steuer, R., Wallenius, J. and Zionts, S. (1992) 'Multiple criteria decision making, multiattribute utility theory: the next ten years', *Management Science*, Vol. 38, No. 645, pp.654– 1992.

Eskandari, H., Rabelo, L. and Mollaghasemi, M. (2005) 'Multi-objective simulation optimization using an enhanced genetic algorithm', in Kuhl, M., Steiger, N., Armstrong, F. and Joines, J. (Eds.): *Winter Simulation Conference*.

Goldberg, S. (1958) *Introduction to Difference Equations*, John Wiley & Sons, Inc., New York, New York.

Harrison, G. (1979) 'Global stability of food chains', *The American Naturalist*, Vol. 114, No. 3, pp.455–457.

Kleijnen, J.P.C. (1975) 'Screening designs for poly-factor experimentation', *Technometrics*, November, Vol. 17, No. 4, pp.487–493.

Korhonen, P. (1988) 'A visual reference direction approach to solving discrete multiple criteria problems', *European Journal of Operational Research*, Vol. 34, No. 2, pp.125–159.

Korhonen, P. and Wallenius, J. (2008) 'Visualization in the multiple objective decision-making framework', in J.B. et al. (Eds.): *Multiobjective Optimization*, pp.195–212, Springer-Verlag, Berlin.

Korhonen, P., Yu, G. and Macdonald, G. (1997) *Quadratic Pareto Race*.

Kuriger, G. and Grant, F. (2011) 'A lexicographic Nelder-Mead simulation optimization method to solve multi-criteria problems', *Computers & Industrial Engineering*, Vol. 60, No. 4, pp.555–565.

Kuriger, G. and Ravindran, A. (2005) 'Intelligent search methods for nonlinear goal programming problems', *INFOR*, Vol. 43, No. 2, pp.79–92.

Lee, E. and Messerschmitt, D. (1987) 'Synchronous data flow', *Proceedings of the IEEE*, Vol. 75, pp.1235–1245.

Lee, E. and Sangiovanni-Vincentelli, A. (1998) 'A framework for comparing models of computation', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 12, pp.1217–1229.

Mauro, C. and Smith, D. (1984) 'Factor screening in simulation: evaluation of two strategies based on random balance sampling', *Management Science*, February, Vol. 30, No. 2, pp.209–221.

Montgomery, D. (1979) *Factor Screening Methods in Computer Simulation Experiments*, Technical Report, Georgia Institute of Technology.

Myers, D. (2013) *A Multi-Criteria Policy Set Optimization Framework for Large Scale Simulation Models*, PhD thesis, University at Buffalo, The State University of New York.

Nelder, J. and Mead, R. (1965) 'A simplex method for function minimization', *Computer Journal*, Vol. 7, No. 4, pp.308–313.

Satterwaite, F. (1959) 'Random balance experimentation', *American Society for Quality*, Vol. 1, No. 2, pp.111–137.

Spendley, W., Hext, G. and Himsworth, F. (1962) 'Sequential application of simplex designs in optimization and evolutionary operation', *Technometrics*, Vol. 4, No. 4, pp.441–461.

Tauer, G. (2009) *A Graph-Based Factor Screening Method for Synchronous Data Flow Simulation Models*, Master's thesis, May, Rochester Institute of Technology.

Tauer, G., Kuhl, M., Costantini, K. and Sudit, M. (2010) *A Heuristic Screening Method for Graph-Based Simulation Models*, Technical Report, January, Rochester Institute of Technology [online] http://people.rit.edu/mekeie/TaueretalJOC10.pdf (accessed 3 August 2015).

Wallenius, J., Dyer, J., Fishburn, P., Steuer, R., Zionts, S. and Deb, K. (2008) 'Multiple criteria decision making, multiattribute utility theory: recent accomplishments and what lies ahead', *Management Science*, July, Vol. 54, No. 54, pp.1336–1349.

Wangersky, P. (1978) 'Lotka-Volterra population models', *Annual Review of Ecology and Systematics*, Vol. 9, No. 1, pp.189–218.