Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

SUPPLY CHAIN MULTI-OBJECTIVE SIMULATION OPTIMIZATION

Jeffrey A. Joines
Deepak Gupta

Department of Textile Engineering
North Carolina State University
Raleigh, NC 27695-8301, U.S.A.

Mahmut Ali Gokce
Russell E. King
Michael G. Kay

Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906, U.S.A.

ABSTRACT

A critical decision companies are faced with on a regular basis is the ordering of products and/or raw materials. Poor decisions can lead to excess inventories that are costly or to insufficient inventory that cannot meet its customer demands. These decisions may be as simple as "How much to order" or "How often to order" to more complex decision forecasting models. This paper addresses optimizing these sourcing decisions within a supply chain to determine robust solutions. Utilizing an existing supply chain simulator, an optimization methodology that employs genetic algorithms is developed to optimize system parameters. The performance measure that is optimized plays a very important role in the quality of the results. The deficiencies in using traditionally used performance measures in optimization are discussed and a new multi-objective GA methodology is developed to overcome these limitations.

1 INTRODUCTION

Competitiveness in today's marketplace depends heavily on the ability of a company to handle the challenges of reducing lead-times and costs, increasing customer service levels, and improving product quality. Traditionally, sourcing (procurement), production, distribution and marketing have been working independently. Unfortunately, although they seem to be working towards a common goal, these organizational units have different objectives. Marketing wants to have a high customer service level as well as high sales volume, but this conflicts with the objective of production and distribution. Sourcing decisions normally depend solely on minimizing the cost of goods, and production and distribution decisions often consider only maximizing throughput while minimizing production (unit) costs without any consideration for high inventory levels or long lead-times. Supply chain management is the effective coordination and integration of different organizations with different objectives towards a common goal. The great potential for improvement in these objectives through effective supply chain management mechanisms has recently been realized (Karabakal et al. 2000, Lyon et al. 2001).

A supply chain, from an operations perspective, has three components: sourcing or procurement, manufacturing and distribution, and inventory disposal. The focus of this paper is on decision making in the sourcing component. In particular, we develop a simulation-based, genetic procedure for determining optimal setting for controllable inputs. Sourcing decisions have a large impact on manufacturing and distribution and inventory disposal as well. Therefore, sourcing and procurement decisions directly affect the efficiency of the entire supply chain. Because sourcing decisions include the supplier, which is usually a separate company, these decisions are much more rigid than manufacturing, distribution, and inventory disposal decisions. Whereas manufacturing and inventory disposal decisions might be internal to a company most of the time and therefore easier to change or modify, sourcing decisions that include outside companies will be hard to change due to contracts and agreements.

Supply chain problems are often very large and complex owing to the interactions between the entities, the length of the supply chain, the lead times of manufacturing and shipping, the complexities of modeling the individual entities, the stochastic nature of the demands, etc. Because of these complexities, very few analytical models exist except for simplified versions of the problem which often are based on limiting assumptions. Even if the analytical forms do exist, it is very difficult to solve these models using traditional search methods like linear programming, differentiation, or even local gradient-based methods owing to the fact that most of the models are discrete, non-linear and/or multimodal. Therefore, heuristic or computational methods are required to even determine good solutions.

Computer simulation is a methodology that can be used to directly model the complexities of the entire supply

chain without the limiting assumptions. It can be used to describe and analyze the behavior of a supply chain and can aid in the design/control of the supply chain through evaluation of "what if" questions (i.e., what if we source from these two suppliers? or what if we drop ship 20% of the estimated demand from one supplier and quick replenish every four weeks from the other supplier?). However, other practical questions (such as Which combination of suppliers is best? and What is the best sourcing strategy under these conditions?) seek optimum values for the decisions variables of the system for the one or more performance measures. In this case, the simulation model can be thought of as an objective function and/or constraint functions in optimizing these complex stochastic systems.

Using simulation in the optimization process presents several challenges. First, there is no analytical expression of the objective function, which eliminates differentiation, or exact calculation of local gradients. Further, the stochastic nature of the simulation causes problems because given a set of deterministic decision variables, the performance measure is not crisp but rather is described by a probability distribution. Simulation programs are typically computationally more expensive to evaluate than analytical functions. Therefore, the efficiency of the optimization algorithms is more crucial.

This paper presents a supply chain simulation optimization methodology employing genetic algorithms. In general, this paper considers at the sourcing of general merchandise from a supplier for a retailer. To test the technique, the Sourcing Simulator (Hunter et al. 1992, Hunter et al. 1996) is used as the simulation tool/evaluation function. The Sourcing Simulator can evaluate a sourcing strategy for a retailer for a given set of inputs. The tool is expanded using the optimization methodology to find the best scenario for a given performance measure/measures.

1.1 Performance Measures

An important issue in supply chain management is how to measure the performance of a supply chain for a given set of decision variables. No matter how appropriate the methodology, if the performance measure is poor, the results could be misleading or false. A key issue is that multiple objective problems are faced frequently in supply chain management (i.e., companies want to optimize more than one criteria that are often conflicting). Typically, the objectives are to maximize revenue with minimal inventory while maximizing customer service. Multi-objective optimization problems are a major challenge in developing solution methodologies. Ideally, one would like to have a single performance measure that addresses all three of these issues. Unfortunately, traditionally used performance measures do not take into account all three. Gross Margin (GM) is a widely-used supply chain performance measure that takes into account the profitability of the company:

$$\begin{aligned} \text{GM} &= \text{Total Revenue (TR)} - \text{Cost} \\ &= \text{TR} - (\#\text{ of units purchased } x \text{ Whole cost/unit}). \end{aligned}$$

GM is a measure of revenue only. It neither takes into account the cost of carrying inventory and nor relates to the chain's customer service level. In an effort to improve GM, Gross Margin Return on Investment (GMROI) adjusts GM for the average inventory held over the period and is an effective measure that takes into account the money earned and inventory held as calculated in the following formula:

$$\text{GMROI} = \frac{\text{GM}}{\text{Average Inventory Cost}}.$$

It can be interpreted as the margin earned per dollar invested in inventory. It is one of the measures that is used to evaluate the effectiveness of solution alternatives in the current decision support system built on top of the Sourcing Simulator. However, it does not consider the customer service level and in earlier studies it is shown to be a poor objective function when used directly in an optimization algorithm (Gokce 2002). GMROI is misleading in that it can be maximized by buying just a very few items, in which case GM will be very low but the average inventory will be even lower. Therefore, GMROI is high but the customer service level is very low since only a little of the consumer demand is met while the revenue is also poor. A new measure (GMROISL) was proposed that incorporates customer service by multiplying GMROI by the in-stock percentage (Gokce 2002). However, this presumes a weighting between in-stock percentage and GMROI which is difficult to determine. Also notice that, in measuring customer service, lost sales are more important than the in-stock %. Minimizing the lost sales directly relates to maximizing customer service, while being out of stock for a SKU (Stock Keeping Unit) does not necessarily decrease customer service level in a particular period if there was no demand for that SKU during that period.

1.2 Sourcing Simulator

The Sourcing Simulator is a stochastic simulation model for the consumer product retailing process developed by Nuttle et al. (1991) and Hunter et al. (1992). The model allows investigation of the effects of alternative retailing procedures on financial and other performance measures for a retail store. The value of the model lies in the fact that it captures the random nature of consumer behavior at the retail store within a robust framework that allows investigation of buyer strategies. Consumer arrivals at the retail store are modeled as a time-dependent Poisson process. The rate each week

is based on a specified season arrival pattern. The model tracks the inventory by SKU. A forecast of consumer demand is expressed in terms of customer volume, SKU mix and presumed seasonality. The model assumes that this forecast is in error. This error is specified as a volume error, SKU mix error, and actual seasonality.

It models alternative mechanisms for supplying product to a retail store. The model tracks the inventory of a line of product offered in a range of SKUs. The store sets up an initial inventory to start the selling season according to the store buyer's plan. Customers arrive at the store and attempt to purchase garments. For a particular customer, if the desired SKU is in stock, a sale is recorded and inventory decremented. If the SKU is out of stock, a stockout is recorded. In either case the customer may look for another item with certain probabilities. The store may issue replenishment orders on the vendor. Replenishment may be based upon the original buyer's plan or may reflect the use of actual Point-of-Sale (POS) data. In this way the selling season is played out and performance statistics are computed. Within the season, the buyer may employ one of two alternative techniques for re-estimating season's demand and incorporating the re-estimate in a scheme for issuing reorders to the manufacturer.

As for the reorders, the number and timing of reorders are specified prior to the beginning of the season. The model allows for planned price reductions (markdowns) in order to stimulate the sales for a specific period or at the end of the season. Consumer behavior is effected in two ways by a markdown. First, customer arrivals to the store increase proportional to the reduction in price based on price elasticity. Second, the probability that a customer selects an alternate SKU after encountering a stockout increases.

Hunter et al. (1996) present several case studies for illustration of re-estimation of demand and quick response (QR) versus traditional retail practice. The case studies showed clearly that useful information can be made available very early in the season both to the buyer and, if shared, to the apparel manufacturer and textile producer, to whom lead-time is critical by use of demand re-estimation. Comparisons of performance parameters for both traditional and QR procedures showed clearly the superiority of the QR methods.

Hunter et al. (1992) expanded the capability of the Sourcing Simulator to include a responsive fabric-supply component by modeling the manufacturer portion to form an apparel-supply chain system for QR retailing. In-season apparel shop order releases are calculated weekly depending on retail orders received but not yet shipped, backorders, work-in-process, finished inventory, availability of fabric and constrained on minimum release batch for individual SKUs, maximum number of SKUs in a shop order and shop capacity. Weekly, in-season fabric reorders are calculated based on net fabric requirements from apparel shop orders. The order size is limited by the allocated weekly apparel shop capacity, as there is no need to carry fabric that cannot be used. Results obtained using the simulation model are also presented in the paper. They conclude that apparel-manufacturing systems exist that allow retailer performance to come very close to the case with a perfect supply by the vendor.

King and Hunter (1996) used the Sourcing Simulator model for demand re-estimation and inventory replenishment of basic apparel in a specialty retail chain. Their results show the sponsor retail chain achieved substantial reductions in inventories while improving customer service levels by adopting the reorder procedures outlined. Hunter et al. (1996) explore the applicability and benefits of QR compared to traditional retailing procedures. They extend their previous work and present results on the impact of assortment error, volume error, price markdowns, reorder lead-time, season length, number of SKUs on performance. King and Maddalena (1998) describe the case of simulation for replenishment analysis with a major retailer.

The Sourcing Simulator can simulate sourcing scenarios that contain a large number of SKUs. The number of decisions that can be made by the analyst is large and even finding good decisions is very hard in a trial an error approach which is what is currently being done by hundreds of companies that use the tool. Owing to the stochastic nature and non-linearity of the problem as well as the fact that some of the variables are discrete, a solution methodology using GAs will be developed.

2 SIMULATION OPTIMIZATION

Law and McComas (2000) define simulation optimization as the "orchestration of the simulation of a sequence of system configurations (each configuration corresponds to particular settings of the decision variables (factors)) so that a system configuration is eventually obtained that provides an optimal or near optimal solution." Several excellent survey papers are written on simulation optimization techniques and procedures. Androdottir (1998) and Fu (2001) present reviews of simulation optimization techniques both for continuous and discrete decision variables. A detailed description of available simulation based optimization packages, their vendors, and the heuristic search procedures that they use may be found in Law and Kelton (2000).

2.1 Genetic Algorithm

Since the supply chain optimization problem is very complex, the solution space is not easy to search owing to the landscape and the size of the space (i.e., stochastic simulation). Genetic algorithms (GAs) are a powerful set of stochastic global search techniques that have been shown to produce very good results for a wide class of problems. GAs

can find good solutions to linear and nonlinear problems by simultaneously exploring multiple regions of the solution space and exponentially exploiting promising areas through mutation, crossover and selection operations (Michalewicz 1996). In general, the fittest individuals of any population are more likely to reproduce and survive to the next generation, therefore improving successive generations. However, some of the inferior individuals can, by chance, survive and also reproduce. Unlike many other optimization techniques, GAs do not make strong assumptions about the form of the objective function (Michalewicz 1996). Whereas traditional search techniques use characteristics of the problem (objective function) to determine the next sampling point (e.g., gradients, Hessians, linearity, and continuity), the next sampled points in genetic algorithms are determined based on stochastic sampling/decision rules, rather than a set of deterministic decision rules. Therefore, evaluation functions of many forms can be used, subject to the minimal requirement that the function can map the population into a totally ordered set. A more complete discussion of GAs, including extensions to the general algorithm and related topics, can be found in books by Goldberg (1989), Holland (1992) and Michalewicz (1996).

Each solution (individual) in the population of a GA is described by its chromosome representation, which is a vector of variables. The first step in a GA is to initialize the population. This can be done either randomly or by seeding. Once the initial population is generated, each individual, i, is evaluated using the objective function to determine its fitness or value, F_i. To parent the next generation, a subset of the population is selected. It is possible for an individual in the population to be selected more than once to be a parent. A probabilistic selection is performed such that the fittest individuals have an increased chance of being selected. These parents then undergo reproduction using genetic operators to produce a new population. The two basic types of genetic operators are crossover and mutation. In crossover, two new individuals are reproduced from two parents by combining the parent chromosomes. In mutation, randomly selected individuals from the population undergo changes in chromosomes randomly. To complete the new population, a subset (arbitrary or otherwise) of the old population is added to the new population. The general GA is summarized in Figure 1.

1. Set generation counter $i \leftarrow 0$.
2. Create the initial population, P_0, by randomly generating N individuals.
3. Determine the fitness of each individual in the population by applying the objective function to the individual and recording the value found.
4. Increment to the next generation, $i \leftarrow i + 1$.
5. Create the new population, P_i, by selecting N individuals stochastically based on the fitness from the previous population, P_{i-1}.
   (a) Randomly select R parents from the new population to form the new children by application of the genetic operators.
   (b) Evaluate the fitness of the newly formed children by applying the objective function.
6. If $i <$ the maximum number of generations to be considered, go to Step 4.
7. Output the best solution found.

Figure 1: A Simple Genetic Algorithm

2.2 Multi-Objective GA

Often optimization problems often have multiple objectives. Most of the time these objectives are conflicting (i.e., optimizing one objective causes the other objectives to be poor). For example, consider a grocery store simulation where one is trying to determine the optimal number of baggers and check out clerks needed during each time period the grocery store is open. Minimizing the overall cost will ultimately lead to only one bagger and clerk for each period while optimizing for the minima customer wait time would lead to n baggers and clerks where n is the number of checkout lines. When only looking at one objective, the other objective suffers. However, in this case both objectives are quite important and they need to be optimized together.

In this paper, the objectives $(f_1, f_2, \ldots f_k)$ are output performances generated from a stochastic simulation, making the search even more difficult. It is very rare that a single optimal point will optimize all of the objectives at once. Therefore the notion of optimal is different than when considering a single objective, the term *Pareto optimal* will be used. A solution (X_1) is Pareto optimal if there does not exist another point $X_2 \in F$ such that $f_i(x_1) \leq f_i(x_2)$ for all $i = 1, \ldots, k$ except $i = j$ and $f_j(x_1) < f_j(x_2)$ for at least one j. A set of *nondominated* solutions is now obtained to generate a Pareto optimal frontier. All points long the frontier are Pareto optimal (as seen in Figure 2).

2.3 Multi-Objective Optimization

Over the past 20 years, many traditional methods have been developed for this problem. Most of these methods are quite limiting in their abilities to solve only linear objectives and constraints. Since the objectives are being generated from a stochastic simulation, the methods appropriate for simulation optimization will have to be used. See Coello Collelo (2002) for a critical review of all evolutionary multiobojective techniques. There are three approaches that one could use to solve the objective value.

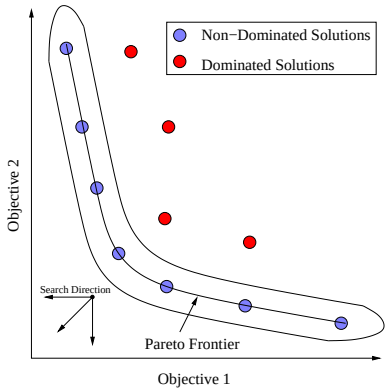

Figure 2: Pareto Frontier for Two Objectives

2.3.1 Decision before Search

Probably the simplest and easiest method is to aggregate the objectives into a singular scalar function

$$g(x) = \sum_{i=1}^{k} w_i f_i(x)$$

where w_i are the weighting functions which can determine the relative importance of each of the objectives and $\sum_{i=1}^{k} w_i = 1$. The relative importance is determined by the utility of the decision maker. Now any of the standard scalar optimization techniques can be employed to solve it (e.g., Stochastic Approximation, OptQuest, etc.). The solution determined represents only one point on the Pareto frontier. Multiple set of weight values can be tried but there is no guarantee the frontier can be generated. Also, determining the relative weights can be quite difficult since the scale of the individual objectives plays a dramatic role. For example, the case of trying to maximize GMROI (a cost metric) and service level percentage, GMROI will dominate the objective to effectively only optimizing GMROI.

Other approaches include goal programming where the decision maker provides targets or goals that they wish to achieve for each objective. The purpose is to try to minimize the absolute deviations among the objective and the targets ($\sum_{i=1}^{k} |f_i(x) - T_i|$). Again, single objective techniques can be used. However, the decision maker has to know about the individual objective ranges and the solution generated will lie in the dominated region if the targets do.

Another more effective method of using thresholds, is to optimize over one objective (f_1) and constrain the other objectives to be within some threshold. For example, one could maximize GMROI while constraining the service level % to be at least 95%. There are two criticisms of this approach for decision making. One, under certain conditions (variable ranges, data) 95% may not be obtainable. Second, this will be a binding constraint and will produce only one point on the frontier.

2.3.2 Search before Decision

When producing only one point, the decision portion has been done before the search process has been performed. In the previous example, the possibility that GMROI could be doubled if the company were willing to accept a 92% service level would be lost in the analysis. Having the frontier allows for a more robust decision to be made since the frontier gives the analyst the ability to see all possibilities. Pareto based GA methods try to generate the entire Pareto frontier which will allow the decision maker to make a decision based on the frontier. Most of the GA Pareto methods modify the selection process in determining which points survive as well as undergo reproduction to generate new points (Coello Coello 2002). The modifications take into account that there are more than one objective.

Vector Evaluated GA (VEGA) by Schaffer was the first multi-objective GA method. It modifies the selection operator by performing proportional (roulette) wheel selection using each objective to select a number of sub-populations. For example, if there are two objectives, half the population will be selected using f_1 and the other half using f_2. Then the sub-populations are shuffled together to form a new population. This is very simple and efficient , but solutions generated are what are called locally nondominated but not necessarily globally nondominated. Individuals excel only along one objective. The frontier created will be mainly cluster near the ends of the frontier (as if we only optimized one objective). More efficient methods like the Niched Pareto GA (Horn et al. 1994) more effectively generate a full Pareto frontier.

Nondominating Sorting GA (NSGA-II) by Deb and Goel (2002) is of the best methods for generating the Pareto frontier and is used in this study. The NSGA-II algorithm ranks the individuals based on dominance. The fast nondominated sorting procedure (Figure 3) allows us to find the nondomination frontiers in Step 4 of the main algorithm where individuals in the frontier set F_1 are not dominated by any solution and those in F_2 are dominated by only those individuals in F_1. Also, it determines a new fitness value (i_{rank}) that represents the frontier number (i.e., all individuals in F_1 are given a fitness of one, F_2 two, etc.).

In Step 6a of the main algorithm (Figure 4), the crowding distance is calculated (seen in Figure 5) for each individual of the new population. Crowding factor gives the GA the ability to distinguish individuals that have the same rank (i.e., those that reside in the same frontier set F_i). This forces the GA to uniformly cover the frontier rather than bunching up at several good points by trying to keep population diversity.

The comparison operator ($\geq_n$) given in Figure 6 is used by the GA to sort the population for selection purposes.

1. For each $p \in P$
   (a) For each $q \in P$
   - If p dominates q then $S_p = S_p \bigcup \{q\}$ which includes q into the set of solutions dominated by p
     Else IF q dominates p then
     $$n_p = n_p + 1$$
   - If $n_p = 0$ then $F_1 = F_1 \bigcup \{p\}$
2. $i = 1$
3. While $F_i \neq \{\}$
   - $H = \{\}$
   - For each $p \in F_i$
     (a) For each $q \in S_p$
     i. $n_q = n_q - 1$
     ii. if $n_q = 0$ then $H = H \bigcup \{q\}$
   - $i = i + 1$
   - $F_i = H$ next frontier has now been formed
4. Return the set of frontiers F_i

Figure 3: Fast Nondominating Sorting

1. Randomly initialize P_0 and set $Q_0 \leftarrow \{\}$
2. $t \leftarrow t + 1$
3. $R_t \leftarrow P_t \bigcup Q_t$ combine parent and children population
4. $[F, i_{\text{rank}}]$ ←fast_nondominated_sort(R_t) where F equals all non-dominated fronts of R_t
5. $P_{t+1} \leftarrow \{\}$
6. While $|P_{t+1}| < N$
   (a) $i_{\text{distance}} \leftarrow$ crowding_distance_assignment(F_i)
   (b) $P_{t+1} = P_{t+1} \bigcup F_i$ include the i dominated set in next population
7. Sort($P_{t+1}, \geq_n, i_{\text{rank}}, i_{\text{distance}}$)
8. Select the top N, $P_{t+1} = P_{t+1}[1 : N]$
9. $Q_{t+1} \leftarrow$ make_new_pop(P_{t+1}) via selection, crossover, and mutation
10. If stopping criteria met then stop otherwise go to 3

Figure 4: NSGA-II Main Algorithm

Individuals that are in a lower domination frontier set are considered better than those in higher sets (i.e., F_1 individuals are relatively better than individuals in F_2). If they are in the same frontier, then the individuals which is the farthest from other individuals is considered better (i.e., this individual fills a gap on this frontier set).

1. $i_{\text{distance}}(j) = 0 \forall j = 1, ..., p$
2. For each objective k,
   - Sort P based on k
   - $i_{\text{distance}}(1) = i_{\text{distance}}(p) = \infty$
     Boundaries are always selected
   - For j = 2 to $p - 1$
     $i_{\text{distance}}(j) = i_{\text{distance}}(j) + (f_m(i+1) - f_m(i-1))$

Figure 5: Crowding Distance Function

- Given the non-domination rank (i_{rank}) and local crowding distance (i_{distance})
- $i \geq_n j$ if ($i_{\text{rank}} < j_{\text{rank}}$ or (($i_{\text{rank}} = j_{\text{rank}}$) and ($i_{\text{distance}} > j_{\text{distance}}$))

Figure 6: Comparison Operator ($\geq_n$)

The NSGA-II main loop can be seen in Figure 4. The first step is the same as Steps 2 and 3 of the general GA in Figure 1. In Step 3, we add the newly generated children (Q_t) to the parent population. Next, the domination frontiers are determined as well as the new fitness value (i_{rank}) for each individual. In Step 6, the next population is selected from the top nondominated frontier sets (F_i) and the crowding distance is determined for P_{t+1}. Next, the population is sorted using the $\geq_n$ operator based on the i_{rank} and i_{distance}. In Step 9, the tradional GA procedures are performed with selection being the only modification.

3 APPROACH & EXPERIMENTATION

The effectiveness of the multi-objective simulation optimization technique is tested on a real data set that is currently being analyzed for a company. As stated earlier, they want to determine the parameter settings that will give them the best GMROI while maintaining a high service level (e.g., 95%). Initially, Gocke (2002) maximized GMROI while constraining the service level to 95%. They then wanted to know the average GMROI if the service level was 94% or 93%. However, they would be willing to live with a lower service level if the GMROI could be greatly improved. Therefore, we decided to modify NSGA-II algorithm to work in this environment to allow them to see the frontier so they could make their own decisions. Table 1 represents the fixed scenario specifics as defined by the company.

Table 1: Scenario Specifics

Planned number of units to sell	4800
Season Length	20 peaks
Seasonality	Mid peak
No. of Styles/Colors/Sizes	2/4/6 (48 SKUs)
Planned Percent	
Style	60/40 %
Color	10/20/30/40 %
Size	5/15/25/30/15/10 %
SKU Mix Error	
Style	40 %
Color	40 %
Size	20 %
Cost Data	
Initial whole sale cost	$12.50
Replenishment whole sale cost	$12.50
Retail price	$25.50
Liquidation price	$8.00
Ordering Cost	$25.50
Initial fixed shipping cost	$100.00
Replenishment fixed shipping cost	$100.00
% who chose alternative	24%
Markdowns	
No. of markdowns	1
Markdown Start Week	18
Markdown Duration	3 wks
Markdown %	25%
% who chose alternatives	50%

3.1 Modified GA

In building a GA methodology to solve the supply chain sourcing problem, six fundamental issues that affect the performance of the GA must be addressed: chromosome representation, initialization of the population, selection strategy, genetic operators, termination criteria, and evaluation measures. In the following subsections, these issues are introduced and described specifically for the proposed multi-objective GA.

Chromosome Representation For any GA, a chromosome representation is needed to describe each individual in the population. Chromosome representation determines how the problem is structured in the GA, as well as the genetic operators that can be used. For the sourcing decision, the chromosome representation in this case is fairly straightforward. An individual is kept as a vector of size 51, where each cell corresponds to a decision variable (as seen in Table 2).

Notice, that not all combinations of the decision variables constitute a feasible solution. For this reason, infeasible solutions are repaired using a repair function before they are evaluated by the Sourcing Simulator.

Initialization of the Population The initial population is formed randomly based on the upper and lower bound for each of the decision variables in a chromosome using a uniform distribution.

Table 2: Decision Variables and Ranges

Decision Variable	# of Variables	Bounds	Type
Initial Drop %	48	[0 100]	Real
Number of Reorders	1	[0 18]	Discrete
Reorder start week	1	[1 18]	Discrete
Min. Order Quantity	1	[1 60]	Discrete

Selection Strategy Selection of parents to produce successive generations is very important in driving the search. The goal is to give more chance to the "fittest" individuals to be selected. For each selection scheme, probabilities are assigned to the individuals. The better individuals have higher probabilities. A normalized geometric ranking scheme is used for the proposed genetic algorithm in this paper. Individuals are first ranked from best to worst according to their fitness values. Then each individual is assigned a probability based on the rank from a truncated geometric distribution (Joines et al. 1996). In the original NSGA-II method, Deb et al. use a tournament selection where the tournament is based on the $\geq_n$ operator. Since the population is sorted from best to worst in Step 7, the normalized geometric ranking scheme does not require any more sampling or sorting as does the tournament selection.

Genetic Operators Reproduction is carried out by application of genetic operators on selected parents. Four mutation (Boundary, Uniform, Nonuniform, and Multi-NonUniform) and three crossover operators (Simple, Arithmetic, and Heuristic) are used based on the representation. Continuous variables use the version by Michalewicz (1996) while the discrete variables use the modifications by Joines et al. (1996).

Termination Criteria The GA is terminated after a specified number of generations.

Evaluation Measure Genetic algorithms rely on the simple premise of using natural selection as a means of solution elimination. The objective function is the driving the force of the GA search. In this research, instead of performing an analytical function evaluation, each solution is simulated to determine its performance. Because the simulation is based on a particular forecasted demand level and seasonality (i.e., nonhomogenous possion process of customers arriving to the store), the answers generated need to be as robust as possible. Therefore, we performed 25 replications at each of the following volume errors (i.e., –20, –15, –10, –5, 0, 10, 15, and 20%) where –20% error means that the true demand will be 20% under what was planned. An overall average and standard deviation across all error levels and replications for GMROI and service level % is used as the objective value for those input parameters. Each of the replications uses common random numbers.

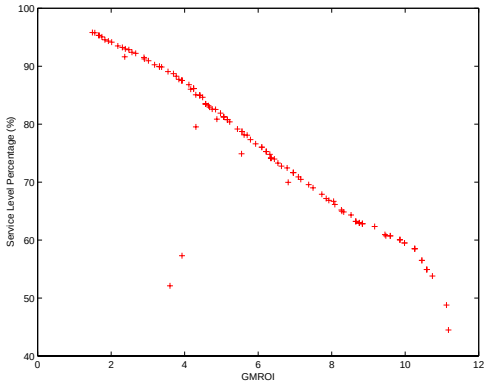

Figure 7: Pareto Frontier for 300 Generations

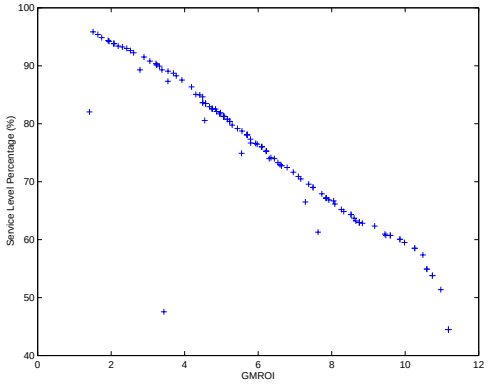

Figure 8: Pareto Frontier for 500 Generations

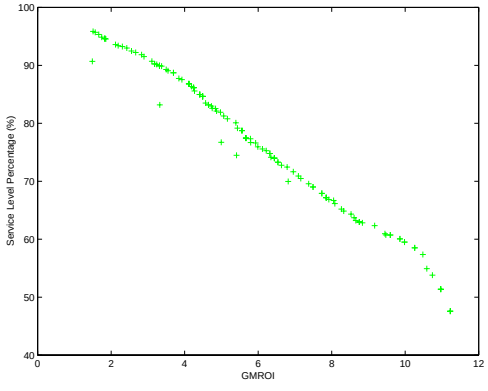

Figure 9: Pareto Frontier for 700 Generations

3.2 Implementation and Results

The Sourcing Simulator code was linked with the GA Optimization Toolbox (GAOT) in MATLAB developed by Joines et al.(1995). This allows the methodology to be tested before embedding it into the decision support system and Fortran simulation code. The algorithm was run for 300, 500, 700 generations and the Pareto Frontiers generated can be seen in Figures 7, 8, and 9. As can be seen the GA does a nice job of generating the frontiers. The longer the GA is run, the more evenly the frontier is covered. Using a 700 MhZ laptop, 300 generations took approximately 20 minutes while 700 generations took 48 minutes.

4 CONCLUSIONS

For this data, the service level of 95% produced a GMROI of 1.5 while a 92% service level produced a GMROI of 2.75. It can be seen that the GMROI can be doubled with a reduction to a 90% service requirement. The frontier allows the analyst to make the best decision. In some cases, the GMROI has not changed much based on decrease in service level. The scenario specific results constrained at 95% service level might not be the best option for the retailer. For a fast moving garment one might wish to maintain a 99% service level. For slower moving items, one might be willing to run out of stock, lowering the service level and spending less on inventory. Because the optimization routine works well, the simulation code is being expanded to allow for different service levels based on volume demand. In the current scenario, the number of weeks of supply stored in inventory was constant across SKUs. Currently, investigating optimizing what the weeks of supply should be for each SKU. Also, the robustness of the algorithm needs to be verified (i.e., the number of replications).

REFERENCES

Androttir, S. 1998. *Handbook of simulation*, Chapter Simulation Optimization, 307–334. John Wiley and Sons.

Coello Coello, C. 2002. *Evolutionary optimization*, Chapter Evolutionary Multi-objective Optimization:A Critical Review, 117–146. Kluwer Academic Publishers.

Deb, K., and T. Goel. 2002. *Evolutionary optimization*, Chapter MEA for Engineering Shape Design, 147–176. Kluwer Academic Publishers.

Fu, M. 2001. Simulation optimization. In *Proceedings of the 2001 WSC*, ed. B. Peters, J. Smith, D. Medeiros, and M. Rohrer, 53–61: Institute of Electrical and Electronics Engineers.

Gokce, M. 2002. *Supply chain optimization*. Ph.d., N.C. State University.

Goldberg, D. 1989. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

Holland, J. 1992. *Adaptation in natural and artificial systems*. 2nd ed. Ann Arbor: The University of Michigan Press.

Horn, J., N. Nafpliotis, and D. Goldberg. 1994. A niched pareto ga for multiobjective optimization. In *Proceedings Congress on Evolutionary Computation*, 82–87: Institute of Electrical and Electronics Engineers.

Hunter, N., R. King, and H. Nuttle. 1992. An apparel-supply system for qr retailing. *Journal of Textile Institute* 83 (3): 462–470.

Hunter, N., R. King, and H. Nuttle. 1996. Evaluation of traditional and quick-response retailing procedures by using a stochastic simulation model. *Journal of Textile Institute* 87 (1): 42–54.

Joines, J. A., C. R. Houck, and M. G. Kay. 1995. A genetic algorithm for function optimization: A Matlab implementation. Technical Report NCSU-IE Technical Report 95-09, North Carolina State University.

Joines, J., C. Culbreth, and R. King. 1996. Manufacturing cell design: An integer programming model employing genetic algorithms. *IIE Transactions* 28 (1): 69–85.

Karabakal, N., G. Gunal, and R. W.. 2000. Supply chain analysis at volkswagen of america. *Interfaces* 30 (4): 46–55.

King, R., and N. Hunter. 1996. Demand re-estimation and inventory replenishment of basic apparel in a specialty retail chain. *Journal of the Textiles Institute* 87 (1): 31–41.

King, R., and R. Maddalena. 1998. Replenishment rules. *Bobbin* 39 (9): 55–56.

Law, A., and W. Kelton. 2000. *Simulation modeling and analysis*. McGraw-Hill.

Law, A., and M. McComas. 2000. Simulation-based optimization. In *Proceedings of the 2000 WSC*, ed. J. Joines, R. Barton, K. Kang, and F. Fishwick, 46–49: Institute of Electrical and Electronics Engineers.

Lyon, P., R. Milne, and R. Orzell. 2001. Matching assets with demand in supply chain management at ibm microelectornics. *Interfaces* 31 (1): 108–124.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd ed. AI Series. New York: Springer-Verlag.

Nuttle, H., R. King, and N. Hunter. 1991. A stochastic model of the apparel retailing process for seasonal apparel. *Journal of the Textiles Institute* 82 (2): 247–259.

AUTHOR BIOGRAPHIES

JEFFREY A. JOINES is an Assistant Professor of Textile Engineering at NC State University. He received his B.S.I.E. B.S.E.E, M.S.I.E., and Ph.D. from NC State University. He recieved the 1997 Pritsker Doctoral Dissertaion Award from IIE. His research interests include evolutionary optimization, object-oriented simulation, and supply chain management. His email address is <JeffJoines@ncsu.edu> and his web address is <http://www.te.ncsu.edu/joines/>.

DEEPAK GUPTA is a graduate student in the Department of Textile Apparel, Technology, and Management at N.C. State University. His interests are in supply chain management.

MAHUMUT A. GOKCE is a graduate student in Industrial Engineering. He received his Master's Degree in Industrial Engineering from N.C. State University, and his areas of interest include demand activated manufacturing, supplier quality management, and supply chain optimization.

RUSSELL E. KING is Professor of IE at NC State University. He received a Bachelors degree in Systems Engineering and a Masters and Ph.D. in IE from the University of Florida. His research interests include control and scheduling of production systems and demand activated manufacturing/ supply systems. His email address is <king@eos.ncsu.edu> and his web address is <http://www.ie.ncsu.edu/king/>.

MICHAEL G. KAY is an Associate Professor of Industrial Engineering at NC State University. He received his B.A. in Economics and M.S. in Industrial and Systems Engineering from the University of Florida and his Ph.D. in IE from NC State University. His research interests include logistics, metaheuristics, material handling, and facilities design. His email address is <kay@ncsu.edu> and his web address is <http://www.ie.ncsu.edu/kay/>.