Reprinted from

Decision Support Systems
The International Journal

Decision Support Systems 14 (1995) 313–327

Triangulation in decision support systems: algorithms for product design

P.V. (Sundar) Balakrishnan [a], Varghese S. Jacob [b,*]

[a] *Bothell Business Administration Program, University of Washington, Canyon Park Business Centre, 22011 26th Ave. SE Bothell, WA 98021 USA*

[b] *Department of Accounting and MIS, Max M. Fisher College of Business, The Ohio State University, 1775 College Road Columbus, OH 43210-1399 USA*

ELSEVIER

ELSEVIER

Decision Support Systems 14 (1995) 313–327

Decision Support Systems

Triangulation in decision support systems: algorithms for product design

P.V. (Sundar) Balakrishnan [a], Varghese S. Jacob [b,*]

[a] *Bothell Business Administration Program, University of Washington, Canyon Park Business Centre, 22011 26th Ave. SE Bothell, WA 98021 USA*

[b] *Department of Accounting and MIS, Max M. Fisher College of Business, The Ohio State University, 1775 College Road Columbus, OH 43210-1399 USA*

Abstract

Often complex decision problems requiring decision aids, such as a Decision Support System (DSS), do not have solution procedures that can generate an optimal solution in a realistic time period. This has led to the specification of heuristic solution procedures. However, the quality of the solution obtained using a heuristic in specific instances can be uncertain and may be open to debate. One approach to increase the confidence in the quality of the obtained solution is to use the triangulation approach recommended and often used in the social sciences. Thus, the result obtained with a specific heuristic can be considered 'good' (i.e., close to optimal) if that result is in the ball park of the result obtained through a maximally different method. In other words, using very different solution techniques helps provide benchmarks and thus enables the decision maker to avoid those solutions which are caught in local maxima. Based on this notion we have designed a prototype GENEtic algorithms based decision support SYStem (GENESYS) for the product design problem. The DSS provides three different solution techniques, specifically, complete enumeration (optimal solution) for small problems, heuristic dynamic programming and genetic algorithms, to address the product design problems.

Keywords: Dynamic programming; Genetic algorithms; Heuristics; Triangulation; Product design; Buyers' welfare

1. Introduction

Decision Support Systems (DSSs) offer a powerful tool for analysing and solving complex decision problems. Given the focus on decision problems a significant portion of the research in the DSS development domain has focused on modelling issues (see Blanning [2], and Chang, Holsapple and Whinston [4] for a review on modelling issues with in DSS). The assumption being, once a model is generated the solution procedure would naturally follow from the formulation. Thus, DSS implementations typically support a modelling and solution approach for a decision problem in a given domain (see for example, Green and Krieger [12], and Couillard, [5]).

Often real life problems requiring DSS aids are complex and NP-Hard. Since determining optimal solutions to these problems are not practical, heuristic solution approaches are used. The quality of the heuristic solution is generally measured by comparing it to a lower or upper bound

* Corresponding author

SSDI 0167-9236(94)00026-3

depending on whether a minimization or maximization type of problem is being solved. However, the solutions generated are tied to the assumptions either explicitly or implicitly made regarding the problem. Thus, a DSS designed for a specific problem that is wedded to a specific modelling and heuristic solution approach may be of limited use. A DSS should, therefore, provide a decision maker the option of choosing a particular solution technique or even better, providing options for solving the problem using different solution procedures and for comparing the obtained results.

In the social sciences, in particular, where the exact solution or value of a particular construct is to be measured, researchers are recommended to use maximally different methods in order to get a better fix on the answer [3]. This concept of using multiple approaches to overcome the problems that may stem from this overt dependence on any one method is known as methodological triangulation [6]. We use this concept as the basis for the design of a DSS for the product design problem. This approach is relevant to situations where it is unlikely that we will ever know the optimal solution to problems of fairly realistic sizes. Consequently. one would have greater confidence in the results if they were generated by alternate approaches. Since, as noted earlier, any one alternate heuristic may fail in particular circumstances. Therefore, one way of checking whether or not the result obtained with a specific heuristic can be considered 'good' (or close to optimal) is if that result is in the ball park of the result using another maximally different approach. In this paper we consider the product design problem and our approach to operationalize the triangulation concept within a DSS context.

The product design problem is a critical one faced by organizations seeking to introduce a new product or refine existing products in the market. Heuristic solutions to the product design problem have been proposed by several researchers in the past (see Green and Krieger, [11] for a review of issues in the domain). In this paper we discuss a prototype DSS which operationalizes the use of the heuristic dynamic programming approach proposed by Kohli and Krishnamurti [17] and the Genetic Algorithm approach proposed by Balakrishnan and Jacob [1]. Additionally the system also includes a complete enumeration technique which can be used for problems of smaller size.

The paper is organized as follows; in Section 2 we discuss the product design problem, its significance and the problems associated with obtaining an optimal solution for the problem. In Section 3, we present the structure of the DSS and discuss the different solution techniques used to solve the problem. In Section 4 a sample interaction with the system is illustrated. The conclusions are presented in Section 5.

2. The product design problem

Several factors such as the high cost of developing and introducing new products, the associated negative impact on organizations of the failure of product acceptance in the market place, make it imperative to design optimal products prior to their launch in an increasingly competitive global market. Therefore, it is not surprising that engineering and marketing researchers spend a considerable amount of time and effort in evaluating and identifying new products before their eventual production.

Before the engineering and manufacture of products, market research of consumer preferences is typically undertaken by most leading organizations in various industries, ranging from automobiles to consumer package goods and services. As a first step in product design the idiosyncratic preference structures of individual consumers are determined by means of conjoint analysis or multidimensional scaling. Once this is done research then turns to address the problem of selecting the appropriate levels of the various attributes to be engineered into the product. For example, for designing a new car one may be interested in determining which combination of attribute levels to have, for example, if 20 mpg, 30 mpg or 40 mpg level are the possible levels for the attribute miles per gallon (mpg) and for the interior of the car, the choices are vinyl, cloth or leather, then there are nine different combinations to choose from.

cess. The end result of the data collection and analysis is a consumers part-worths' matrix. This matrix contains for the representative sample the utility for each level of each of the attributes for each individual consumer. The problem then is one of selecting a specific product to introduce into the market place. The criteria for selecting the product to be introduced typically range from the one that maximizes the market share to that which maximizes the seller's return.

The typical mathematical formulation of the product design problem [20,18] considers that there are a total of K relevant attributes in the product category. Further each attribute has L_k levels, and, there are N consumers in the sample. The product designer based on the consumer utilities data then has to select the specific levels of each of the attributes in order to accomplish the manufacturer's objective. In addition to the matrix of consumer part-worths, the product designer must have information on the competitive environment. In other words, we must be aware of the competitive product offerings in order to design a product that will survive in this market place. In addition, the managerial objective needs to be specified to evaluate the proposed new products. In a typical situation, a firm may be interested in introducing a product into a competitive environment that will result in the largest possible market share (this is referred to as the share-of-choices problem). On the other hand, a public-sector organization may be more concerned with maximizing the consumers' social welfare. In such cases, a *utilitarian function* for selecting a product on the basis of the sum of utilities may be more appropriate [14]. Finding that product profile that results in the largest overall consumer utility is referred to as the buyer's welfare problem.

3. Decision support system structure

The DSS developed for the product design problem is composed of the following main components (Fig. 1):

1. A user interface which is menu driven.
2. A Data base which contains: (a) consumer utilities, (b) details about the competitive environment, namely either a listing of the set of competing products in the market place or a listing of each consumer's status quo product.
3. Model generation and solution strategies.

3.1. The user interface

The user interface is menu driven. Initially it provides the user with the option of choosing a particular solution strategy for solving the problem. Following which the system queries the user for the appropriate files containing the data that forms the input to the system, including the manner in which the problem should be formulated, namely, share-of-choices or buyers welfare (Figs. 2–7 show sample screens).

Further queries are based on the needs of the model generation component for the chosen solution strategy (see Section 4 for a sample interaction with the system). Once a solution has been obtained the system permits one to either quit the session or cycle through the menu again and make a different set of choices.

3.2. The database

The system developed is generic in that it is not tied to any specific product category. Thus, rather than work off of a specific database structure, we work with the concept of data files. The data file names and locations can be specified by the user during the interaction. The only restriction we have imposed is that the user must provide a file of consumer utilities and the number of attributes and attribute levels represented in that file should be consistent with that of the product being designed. Similarly, the number of attributes and attribute levels contained in the file describing information about the competing product set or the consumer's status quo products have to be consistent. This approach allows the user to utilize different files for different competitive product set analysis or for different market segments etc., in a very flexible manner. We first define a product in the form of a binary string to set the stage for the rest of the exposition.

Within the framework of conjoint analysis, a small number of different product profiles are tested, and the idiosyncratic preferences of the individuals are determined for each of the various levels of the different attributes. In the next stage, these individual preference measures (part-worths) are utilized to predict the valuation for any new product profile which was not originally evaluated. Now combining these results across all consumers, a complete enumeration procedure to identify a single product profile that results in the highest share-of-choices may be undertaken [9]. However, in realistic applications, as the number of attributes and attribute levels increase, the number of possible product profiles increases exponentially, and consequently it becomes infeasible even with high speed computers to obtain a realistic solution in a reasonable amount of time. This major limitation has, therefore, led to the specification of heuristics for solving the problem [7,9,15].

An alternative to conjoint analysis is multidimensional scaling [19]. The multidimensional scaling based techniques which have been proposed to address the problem of product design suffers from limitations such as ignoring the problem of cost measurement, and ignoring the cost of technological constraints on the variables ranges [17]. Additionally, these techniques are computationally inefficient except for small problems, an exception to this is the approach of Gavish, Horsky and Srikanth [7] whose heuristics allow solving problems of realistic sizes.

With the growing realization that complete enumeration in the case of selecting a set of product profiles is even more computationally burdensome [10], researchers have started to examine alternate procedures such as dynamic programming, lagrangian relaxation and greedy heuristics for finding 'good' (i.e. close to optimal) solutions. These approaches are, however, not without their limitations as illustrated by Kohli and Krishnamurti [17].

For the purpose of developing this system, we formulate the problem within the conjoint or hybrid conjoint analysis framework. As described earlier, this domain has received much attention and has enjoyed considerable commercial suc-

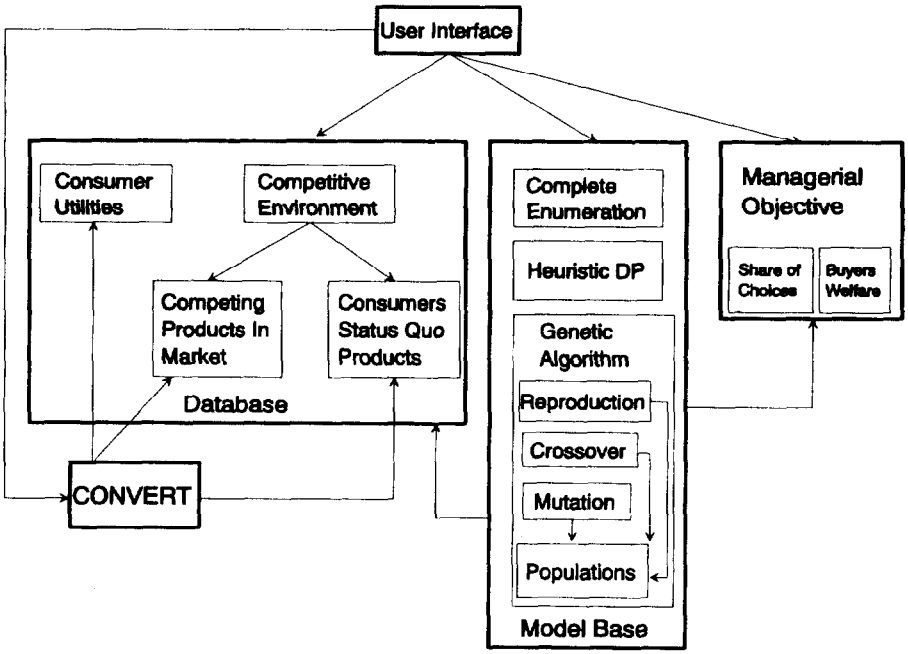

Fig. 1. A Decision support system for product design.

String definition: Given the fixed number of attributes (K) and levels of each attribute (L_k) that exemplify this product category, any product can now be coded as a binary string. Strings can be specified in either dummy variable coding or contrast coding format. If dummy variable coding is employed then generally for each attribute j, $j = 1 \ldots K$, we need $L_j - 1$ variables (or bits) to specify uniquely the presence of a specific level. This implies that we need a total of $\sum_{j=1}^{K}(L_j - 1) = P$ bits to completely specify any product. If on the other hand, a contrast coding scheme is employed, then all L_j bits would be required to represent each attribute. In the contrast coding case, *exactly one* of the L_j bits must be set to one at all times to indicate the presence of a specific level. On the other hand, in the dummy variable representation, *at most one* of the $L_j - 1$ bits can take on the value one as all measurements are made against a base case (specific level).

Consumer utilities

The consumer utilities data file specified by the user is read into a matrix, which we shall call **BETA**. This data is comprised of each consumer's utility for each level of each attribute (i.e. partworth). The data could be obtained as a result of data analysis of consumer preferences done using either dummy variable coding or contrast coding. In the contrast coding case the **BETA** matrix is considered to be the augmented **BETA** matrix as there are as many bits required to represent an attribute as there are levels of that attribute. It is relatively straightforward to convert the data from the augmented **BETA** matrix (i.e. with each attribute described by all L_j levels) to a **BETA** matrix having $L_j - 1$ levels for each attribute. Our implementation contains a procedure (CONVERT) for performing the conversion by a simple rescaling of the consumer utilities as these utilities are invariant to the linear transformation for the fitness criterion employed here. Hence, our DSS is flexible enough in that, irrespective of the data collection and analyses scheme employed by the market researcher, either a **BETA** or an augmented **BETA** matrix can be provided as input. Note that we will assume that there are a total of P levels that are employed to completely specify any product irrespective of the approach used to derive the consumer partworths. As there are N consumers the size of this matrix is N * P.

Competitive environment

The new product under consideration will be introduced into an environment in which there are a number of competitive products. One way of evaluating the proposed new product is to determine how much market share the new product will capture in head-to-head competition with the existing products in the market place. We have operationalized the competitive environment using two different approaches: one approach provides a description of the competing set of product offerings available in this marketplace. It does not specify a status quo product for each individual. The second approach specifies status quo products for each of the N individuals in the representative sample.

Competing products approach: The specification of the competing (set of) product(s) can be based upon managerial judgment and forecasts as to what the likely environment will be at the time of the expected new product launch. The competitive products are described as binary strings and the file in which this data is contained is read into a matrix called **COMP** of size Q * P. Where, P is as defined above, and, Q is the number of competing products in the market place.

A variety of rules can be employed to describe the behaviour of consumers in making their product choice. The impact of the different rules on the final product design is an issue that deserves further investigation. However, here we employ the first choice rule which has been shown to possess excellent predictive validity [13]. The essence of the first choice (or max utility) rule is that consumers will be assumed to choose that product which provides them with the highest utility of all competing products.

One can compute for each consumer ($n = 1 \ldots N$) the utility that would accrue to them from consuming each of the competing products ($q = 1 \ldots Q$). This can be calculated as $\mathbf{BETA}^{*}_{N \cdot P}\mathbf{COMP}^{T}_{P \cdot Q}$. By comparing these utilities (in the resulting N * Q matrix), we can then, for each of the N consumers determine their winning

product from this competitive set. The largest utility value that each consumer obtains from their 'first choice' among the competing products is stored in the matrix $\mathbf{LRGUTIL}_{N*1}$.

Consumer status quo approach: In this approach, following Kohli and Krishnamurti [17] we can arbitrarily assign a status quo product to each consumer based on survey data. These status quo products may be the consumers current favourite brand, or the brand purchased most often from the set of currently available offerings. These idiosyncratic choices are, described as before by means of binary strings, stored in a matrix **ICOMP**. And, as each consumers status quo product is described this matrix is of size N * P. Consequently, as before, one can determine for each individual the utility for his/her status quo product. The utility for the status quo products for each consumer is stored in the matrix $\mathbf{STATQUO}_{N*1}$.

3.3. Model generation and solution

The solution to the product design problem is addressed by either formulating it as a share-of-choices problem, i.e. maximizing the number of customers that will purchase a product over the competitive offerings; or as a buyer's welfare problem, i.e. maximizing the total utility of all buyers; or as a seller's return problem, i.e. maximizing the sellers incremental profit. The current prototype focuses only on the share-of-choices and buyers welfare formulations. The prototype can be easily enhanced to consider the seller's return problem too. However, the current implementation did not focus on it primarily because of the typical difficulty in obtaining the pertinent cost data to address this problem.

Although there are other techniques that could be incorporated into the system, the current prototype makes use of only three solution techniques. These three techniques, however, represent very diverse approaches to solving the problem. Complete enumeration is used to determine the optimal solution for problems of relatively small sizes. Heuristic dynamic programming is essentially a tree search technique used in operations research while Genetic Algorithms (GA) has been classified as a machine learning technique. The GA approach in particular is appealing within the context of this problem as the solution process which is based on the survival of the fittest principle parallels the problem faced by a product that is introduced into the marketplace. We provide below a brief overview of each of the three solution techniques.

Complete enumeration

Each product within this framework can be defined as a specific combination of levels one from each attribute. The various products that can be obtained by the different combinations of attribute levels selected can then be compared on the basis of a managerially pre-specified evaluation function. Complete enumeration is computationally viable only if the product category or service under consideration can be described by a small number of attributes and attribute levels. This procedure enumerates all possible combinations of attribute levels present and the resulting products are evaluated using the pre-specified evaluation function.

In our prototype, we determine the number of consumers that would choose the new product under consideration over their current favourite competitive (i.e., status quo or first choice) product offering as the evaluation criteria for the share-of-choices problem. To address the buyers' welfare problem, on the other hand, the evaluation criteria is modified from merely keeping count of the total number of consumers selecting the new product to keeping score of the total utility accruing from the selection of this new product across all consumers.

Executing the complete enumeration approach requires only a specification of the attributes, their levels and the evaluation criteria. The attribute and levels are obtained from the input files, which provides the consumer part worth utilities and the competitive environment specification while the choice of the evaluation criteria is made at the outset by the user. The system then considers every combination of these levels and evaluates each of the possible products on the pre-specified evaluation criteria (share-of-choices or buyers welfare). The system then iden-

tifies that combination of attribute levels (products) which results in the largest evaluation in a given competitive setting. This analysis can then be rerun by specifying a different competitive set if needed. The specific product that results from this approach is guaranteed to be the optimal solution.

Clearly, as the number of attributes and levels for each attribute increase the number of possible product combinations will correspondingly increase. For example, a product design problem with 8 attributes and 3 levels in each would result in 3^8 or only 6561 different combinations. While a problem with eight attributes and five levels each, the number of possible combinations to choose from increases to 390625 (i.e. more than a 1/3 of a million). Realistic problems typically have substantially more attributes and levels rendering complete enumeration computationally intractable.

Heuristic dynamic programming

This technique was proposed by Kohli and Krishnamurti [17] and further developed by Kohli and Sukumar [18]. Our discussion of this algorithm is essentially based on their work and draws from their description and implementation. The heuristic works by considering attributes as stages and attribute levels as states.

The goal then becomes to identify at the first stage, for each level of attribute 2 the best level of attribute 1. This enables us to identify a number of 'partial' product profiles; where the number of such profiles identified is equal to the number of levels of attribute 2. These profiles are 'partial' as they identify a product by using only a subset of all of the attributes. At the next stage, we identify for each level of attribute 3, the best partial profile, i.e., combination of attributes 1 and 2, from the previous stage. This process is carried on till all of the attributes have been considered. At this point, we are left with a handful of completely specified (i.e., no longer partial) product profiles. At this stage, from this set of completely specified products, we select the best profile. This heuristic essentially uses a build-up procedure in that at each stage another level of an attribute is added on. The selection of the best attribute level is determined by the appropriate choice of evaluation functions, i.e. share-of-choices or buyers welfare.

In our implementation this approach employs the augmented **BETA** matrix. In order to implement the share-of-choices problem, we first convert the consumer utility matrix to a relative part-worths matrix. This conversion is done by simply subtracting from the utility for each level of an attribute the utility of the level of that attribute possessed by that consumer's 'first choice' or status quo brand. This (subtraction) is then carried out across all attributes for all consumers. Let $\text{beta}_{n\ell k}$ denote the part worth of level ℓ $(= 1 \ldots L_k)$ of attribute k for individual n. Let $beta_{n\ell^* k}$ be the utility of the attribute level ℓ^* for attribute k possessed by the status-quo brand for consumer n, then the relative part worth utility of any attribute level ℓ of attribute k for consumer n is:

$$\text{relbeta}_{n\ell k} = \text{beta}_{n\ell k} - \text{beta}_{n\ell^* k}$$

Now, the above relative part-worths matrix can be partitioned into K distinct matrices, RELBETA(k) of size $N * L_k$, each consisting of individuals-by-attribute-levels part worths for each of the $k = 1 \ldots K$ attributes. At the first stage, to each column of RELBETA(1), we add column ℓ $(= 1 \ldots L_2)$ of RELBETA(2) resulting in matrix $\text{SUM}_\ell(2)$. Note there will be, at this stage, as many matrices as there are columns (i.e., attribute levels) in attribute 2. Now for each column in $\text{SUM}_\ell(2)$, we count the number of individuals with utility values strictly greater than zero for the share-of-choices problem. For the buyers' welfare problem, we use the column total of all positive elements. Next, the column with the highest evaluation in each of the (L_2) matrices is identified and selected. The selected columns then form matrix SUM*(2) of size $N * L_2$. At this stage, we have identified a partial profile of only two attributes, where for each level of attribute 2, a level of attribute 1 is selected.

Now, in step 2, to each column of SUM*(2), add column ℓ $(= 1 \ldots L_3)$ of RELBETA(3) resulting in matrix SUM*(3). Following the procedure described above, we recursively perform the steps till we identify the matrix of relative utilities

for a set of full profile products for each consumer in SUM*(K). From this matrix, we can now identify the product that will satisfy the objective by selecting the column with the highest evaluation.

Genetic algorithms

The concept of Genetic Algorithms (GA) was first proposed by Holland [16]. The basis for the algorithm was the observation that a combination of sexual reproduction and natural selection allows nature to develop living species that are highly adapted to their environment. The basic approach, therefore, operates in the following manner.

Algorithm process:. Generate an initial set of products (M), which are represented as strings. Here we employ a binary representation to be consistent with the use of dummy variable coding employed in the conjoint analysis framework to indicate the absence or presence of a specific attribute level). This set forms the initial pool or population of strings of size M.

The following steps are then performed iteratively:

1. Evaluate the fitness of the set of products, in the pool based on some managerially predefined evaluation criteria (e.g. market share, minimize costs, etc.).
2. If stopping criteria met, then stop, else continue.
3. Use various kinds of genetic operators on the strings in the pool to construct a new pool (i.e. set of products) of size M.

This process of evolution and reproduction continues until some pre specified *stopping criteria* is met. At this point, the pool consists of a set of 'highly fit' products.

The genetic operators used to generate new offspring (i.e. products) are:

1. *Reproduction*: A subset of the products in the population of size M is chosen based on their fitness and copies of their strings are generated.
2. *Crossover*: Pairs of reproduced strings are chosen and along specific positions on the strings information between the two strings are exchanged leading to two new strings.
3. *Mutation*: During this process, strings are randomly chosen from the population based on the mutation probability and the value at a specific location in the string is modified.

The technique basically exploits the fact that a 'good' string contains some features (sub-strings) that are desirable and hence contribute to its being evaluated highly. When one exchanges information or features between two good strings the expectation is that it will frequently produce a string that combines the good features of the parents. Thus, as the iterative process continues, the population is going to get fitter strings in it. An extensive discussion on GAs and their applications can be found in Goldberg [8].

The description of applying GA to the product design problem proposed by Balakrishnan and Jacob [1] is now discussed. Genetic Algorithms require the product definition to be in the form of strings. Thus the model generation module generates a population of strings each of which corresponds to a specific product as follows. For the sake of brevity, we describe only the procedure employing the standard dummy variable coding technique. Specifically, as noted before, if there are K attributes in the product category and each attribute j (= 1 ... K) has L_j levels, then a string will be defined by $\sum_{j=1}^{K}(L_j - 1) = P$ locations (or bits) assuming that the attributes are categorical in nature. These P locations, can now be considered to be made up of K sub-strings and the length of a sub string j would be $L_j - 1$. Thus, for example, if we had a product category represented by three attributes A1, A2 and A3, and attribute A1 had three levels, attribute A2 had four levels and attribute A4 had three levels, then an instance of a product string would be, 01 100 01.

Once the product category is defined, i.e. the number of attributes and levels, the system generates an initial population. In other words the system generates instances of strings (i.e. products within the category) to make up a population the size of which is specified by the user (say M).

The initial population is generated so that it contains as diverse a representation as possible. The population is maintained in a matrix $\mathbf{POP}_{M \cdot P}$.

Evaluation: The solution process here is an iterative process, in which the system first evaluates each string in the initial population. The evaluation is done as follows:

1. Determine for each product the utility associated with it by each customer in the **BETA** matrix, so if **PRODUTIL** is the matrix in which this information is stored, then $\mathbf{PRODUTIL}_{N \cdot M} = \mathbf{BETA}^{*}_{N \cdot P} \mathbf{POP}^{T}_{P \cdot M}$.
2. For each product m (= 1 ... M) determine the number of consumers who would choose it over their status quo (or 'first choice') product, i.e. each column in **PRODUTIL** is compared with **STATQUO** (or **LRGUTIL**). If the utility associated with the new product m is greater than that of the current favourite product for a consumer than the new product would be chosen over the competition. By counting the number of consumers who would choose the new over the competition we have a measure of the number of consumers who would switch to this new product, m. This count is used to specify the fitness of a specific product (string).

Once each string is evaluated then the reproduction, crossover and mutation operators are applied, resulting in a new generation of strings which are again evaluated as described above.

Application of GA operators:. Reproduction: The current implementation allows reproduction to take place in either one of the following approaches:

(a) Selection of the 's' (s < M) strings for reproduction from the previous population is done randomly.
(b) Selection of the strings to reproduce is based on picking 's' (s < M) fittest strings. In other words the 's' strings with the highest evaluation scores are chosen for reproduction.

Crossover: The reproduced strings are now candidates for crossover. However, for a string to be feasible, each sub-string in it can have at most a single one in it. In other words each attribute can have only one level active in a product. Thus, suppose we have two feasible strings represented as:

A1 A2 A3

10 001 00 (1)

01 100 01 (2)

Now consider a simple crossover mechanism where all characters are swapped between two locations. Consider for example, a swapping between the third and fourth column positions of strings 1 and 2, resulting in the strings illustrated below:

10 101 00 (3)

01 000 01 (4)

Note that now string 3 is an infeasible string, since for example, if attribute 2 is engine type of a car we are saying that the engine is both, say, a six cylinder and four cylinder engine.

To guarantee a feasible solution one can define crossover as taking place between all locations of an attribute (sub-string). Thus, for example, if attribute 2 is used for crossover, we will have the resulting two strings:

10 100 00 (5)

01 001 01 (6)

Note that both these strings are feasible.

Thus, within the pool of reproduced strings we randomly pick pairs of strings for mating and crossover. The number of attributes for which crossover should occur is specified by the user. The user can specify the number of attributes, r, where $0 < r < K$. The system randomly chooses the r attributes from the total of K that will participate in the crossover.

The new strings which are generated are then evaluated using the procedure outlined earlier. The result of reproduction and crossover is the creation of a new population with strings which may not have existed in the previous population. This new population then forms the basis for the generation of the next population.

Mutation. Once a new population is created, from amongst the members, we provide the strings a chance to mutate. The mutation probability is set

by the user and mutation is performed keeping in mind that feasibility has to be maintained. The location for mutation is randomly picked and the mutation process essentially changes the value of a specific location (bit) in a sub-string (attribute) from a one to a zero or a zero to one. Note that if a zero is converted to one and the sub-string already has a one in a second location, then to maintain feasibility, this second one is converted to zero. Thus, if string 5 shown above (10 100 00), is mutated by turning the second location in A2 from zero to one, it implies that the first location will correspondingly be changed to a zero, thus, resulting in the new string, 10 010 00.

In our implementation, strings are randomly chosen without replacement from the population with some probability (mutation rate). Then a *single* attribute is randomly picked and the mutation process essentially acts to change the level of that attribute resulting in an entirely different candidate string.

The process described above is unlike the typical GA approach, wherein *every bit* position of every string has a probability (i.e., the traditional mutation rate) of undergoing a random change. Therefore, in order to ensure that the expected number of mutations per generation in our implementation is comparable with the traditional approach we have to set our mutation rate equal to the product of the traditional mutation rate value and the length of the string.

Population maintenance. One performs population maintenance to keep the size (M) of the number of strings in each generation a constant. Currently we employ three strategies for population control. The *Emigration Strategy* in which a new population of size M is defined as the collection of (s) reproduced strings and $(M - s)$ offspring strings generated as a result of crossover of the reproduced strings. The second strategy is a *Malthusian Strategy*. Here the size of the old population first grows to include the offsprings produced. This increased size of the population is then subjected to 'natural disaster' culling it down to its sustainable size (M). In our implementation, a set of (s) selected strings from the old population are earmarked as having earned the right to mate and as a consequence by crossover they produce (α) offsprings. Therefore, the old population (M) grows by the number of offsprings (α) generated as a result of the crossover resulting in a population of size $M + \alpha$. From this larger population, the weakest α strings are culled to obtain a new generation of size M. The third strategy implemented is termed a *Modified Malthusian Strategy*. It is a minor variation of the above in that the old population first grows to include all strings from the previous generation (M), plus a copy of the set of reproduced strings (s) and their offspring strings (α) generated as a result of crossover of reproduced strings. Then this larger population $(M + s + \alpha)$ is culled down resulting in a new generation of sustainable size (M). The advantage of this modification being that additional copies of the best strings are maintained and the disadvantage being that some variability in the strings in the final generation is sacrificed.

The process of creating new generations and evaluating them continues till a stopping condition is met. Either a user specified fixed number of iterations or a minimal percentage change in the average fitness of a pre-specified number of (S) best strings over the previous (X) generations can be used as the stopping criteria. Both S and X are specified by the user. When this process terminates it displays as results the collection of strings or products along with their associated evaluation. In addition, the intermediate process results can also be viewed to obtain a large number of strings with very good fitness. The managerial decision-makers then can impose their own preferences such as the strategic fit, costs and technological considerations in selecting the product to introduce.

4. Sample interaction

In this section we illustrate a sample interaction with GENESYS (A GENEtic algorithm based decision support SYStem) for a relatively small sized problem with four attributes and four levels. This small sized problem enables us to determine the optimal solution and hence we can

Table 1
Product Category Description

Attribute	*Colour*	*Size*	*Quantity Pack*	*Scent*	*Shape*	*Package Colour*
Levels	White	2.5oz	Single	Fruity	Rectangle	Beige
	Yellow	3oz	Double-pack	Flower	Cylinder	Gold
	Pink	3.5oz	Triple-pack	Regular	Cube	Silver
	Blue	4oz	Quadruple-pack	No scent	Oval	Black
	Translucent	4.75oz	Six-pack	Antiseptic	Square	Red

compare the quality of the heuristic approaches. Let us assume that the attributes of this product category are colour, size, number of bars in a package scent and package colour for the design and marketing of a bar soap. Table 1 specifies the levels of each attribute. Simulated data was gen-

```
                    Welcome to GENESYS:
A GENEtic Algorithm Based Decision Support SYStem For Product Design
-------------------------------------------------------------------------------

You can choose any one of the following approaches for determining product design.
Note: Complete enumeration is extremely time consuming for problem sizes larger than
eight attributes with three levels each.

1) Complete Enumeration
2) Heuristic Dynamic Programming
3) Genetic Algorithms
4) Exit Program

Enter your choice ( ? for help) : 3
```

Fig. 2. Screen specifying choice of solution techniques.

```
Welcome to the Genetic Algorithm Solution Approach
----------------------------------------------------------

File name containing customers' part worth utilities  :  hbeta.012
File name containing product specification
of current competition                                :  hcomp.012
Enter file name where "Best" product
designs from each generation are to be saved          :  gares.012

Select problem objective
a) Maximize Share of Choices
b) Maximize Buyers Welfare

Enter your choice ( ? for help) : a

Select your Competitive Environment
1) Status Quo Approach
2) Competitive Product Set Approach

Enter your choice ( ? for help) : 1
```

Fig. 3. Screen requesting data files and managerial objectives.

erated for both consumer preferences and the competitive environment [17]. This data was then stored in files which can be accessed by the DSS.

The first step in the interaction with the DSS is choosing a particular solution approach. After the initial opening screens which briefly describe the goal of the DSS, the user is provided with a menu specifying the three available choices for solving the product design problem, namely complete enumeration, heuristic dynamic programming and GA (Fig. 2). Once a choice is made, the system then queries the user for specific information needed by the approach to generate a solution. In the sample interaction shown in Fig. 2, the GA approach was chosen.

Fig. 3 indicates the next step in the GA process, namely a determination of where the consumer preferences and competitive environment data can be accessed from and where to store the results of the run. In addition, in Fig. 3 the elicitation of the managerial objectives are shown, the two main factors being a specification of the problem objective, namely, maximizing share-of-choices, or buyer's welfare as well as the competitive environment to be considered, i.e., the status quo or competitive product set.

The next series of queries deal with setting the parameters for the GA approach. The user has the option of choosing the specific parameters of the reproduction, crossover, mutation and population control strategies (Fig. 4). Note that the maximum value for the number of attributes to crossover is based on the number of attributes being considered in the product category, based on the information provided in the data files. The system automatically specifies this value here. Similarly, once the choice of the size of the genetic pool is provided by the user, the system automatically specifies the maximum number of strings that the user can specify for reproduction.

```
Genetic Algorithm in Use
------------------------

Enter the size of genetic pool [1.. 400]: 100

Select Reproduction Rule:
a) Random reproduction
b) N-Best reproduction

Enter your choice ( ? for help ): b

Enter number of attributes to crossover[1.. 5]: 3
Enter number of strings for reproduction[2.. 100]: 50
Please enter the probability of mutation[0% .. 100%]: 30

Select your Population Maintenance Mechanism
a) Emigration Strategy
b) Malthusian Strategy
c) Modified Malthusian Strategy

Enter your choice ( ? for help ): c
```

Fig. 4. Screen requesting GA parameters.

Fig. 5 illustrates the choice of the stopping condition and the parameters that need to be set for the degree of improvement rule. Given the number of factors that have to be set, as shown in Fig. 6, the system presents the user with the settings specified earlier by the user and provides the user the option to change any setting before it proceeds to execute the model.

Once the solution is generated the user has the option of viewing (see Fig. 7) the result summary of the current execution of the model, or the detailed results in the output file, executing the GA procedure by changing any of the current settings (option 2 takes one back to the screen shown in Fig. 6) or going back to the main menu and using one of the options there.

Table 2 provides the results obtained using the different solution techniques. Since the GA approach generates a population of products the user has the option of specifying the number of 'good' strings to be saved to the results file in each iteration. We have shown the three best strings generated at the end of eleven iterations at which point the process terminated. One of the strong points of the GA approach is that it generates a set of solutions, thus allowing one the flexibility to choose a solution that appeals to the decision-maker on the basis of other strategic criteria.

In this case, the evaluation of the best string by the heuristic DP approach was 61 as opposed to 68 for GA. That is, the DP heuristic is roughly within 90% (89.7%) of the best solution identified by the GA approach. In larger problems for which the complete enumeration procedure is in-

```
Genetic Algorithm in Use
------------------------------

Enter the number of best strings from each
generation that you would like saved ----------> 3

Please enter the stopping condition:
1) Number of iterations rule
2) The degree of improvement rule

Enter your choice ( ? for help ): 2

How many previous (X) generations do you want to
consider for increase in average improvement before
stopping [2.. 10]. Enter X: 5

How many of the (S) best strings from each of the previous X generations
should be considered when checking for the improvement[1.. 100]: 3

The program will stop when the percentage of
difference between the average of the latest
generation and the previous (X) generations is
less than (greater than 0%) : 0.2
```

Fig. 5. Screen requesting stopping condition specifications.

```
GA Specifications Chosen
------------------------------
A) File name containing customer's utility ----------> hbeta.012
B) File name containing product definition
   of current competition ----------> hcomp.012
C) File name where `Best' product designs
   from each generation are to be saved ----------> gares.012
D) Problem Objective (Maximize) ----------> Share of Choices
E) Competitive environment ----------> Status Quo
F) The size of genetic pool ----------> 100
G) Reproduction Rule ---------- > N-Best
H) Number of strings for reproduction ----------> 50
I) Number of attributes to crossover ---------- > 3
J) Probability of mutation ----------> 30.00%
K) Population maintenance ----------> Modified Malthusian Strategy
L) Stopping rule ----------> Improvement rule
N) Number of previous generations to be considered in
   determining average improvement before stopping ----> 5
O) Number of the (best) strings to consider for
   checking the improvement ----------> 3
P) The percentage of difference between the
   average of the latest generation and the
   previous generations that determine the
   stopping condition of the program ----------> 0.200%
T) Maximum number of generation (none= -1) ----------> 100
U) Number of best strings to be saved ----------> 3

[Enter an Option to modify, S to start, Q to quit]

Enter your choice : S
```

Fig. 6. Screen confirming choices.

Table 2
Sample Results

Method	Product Description	Number of consumers choosing the product
GA	Yellow, 4oz., Triple-pack, Fruity, Oval, Silver	68
	Yellow, 3oz., Triple-pack, Fruity, Cylinder, Black	63
	White, 4oz., Triple-pack, Fruity, Oval, Silver	61
DP	White, 4oz., Triple-pack, Fruity, Oval, Silver	61
Optimal	Yellow, 4oz., Triple-pack, Fruity, Oval, Silver	68

What do you want to do next?
0) Result summary
1) Look at the results in the output file
2) Continue GA
3) Go To Main Menu
Enter your choice : **0**

Beta File: hbeta.012
Comp File: hcomp.012
No of Iterations: 11
Evaluation: 68
The Best String: 1000 0010 0100 0000 0010 0100
Elapsed CPU Time: 18500 msecs

Fig. 7. Example of a results screen.

tractable, the use of two very different approaches being fairly close (i.e. 90% of each other) is encouraging to the user of the DSS. On the other hand if we had only one procedure (either DP or GA) the user would not know how to evaluate the fitness of 61 in vacuum and would be somewhat less confident that this solution is relatively close to the optimal value and not caught in some local maxima. The use of maximally different methods has thus permitted us to get a better fix on the solution and thereby increasing the user confidence in the DSS.

5. Conclusions

A DSS to be of value to a decision-maker needs to provide the user with the confidence that the obtained results are indeed of value. Unfortunately, in most instances, the decision aids employ only one particular methodology or approach to tackle a specific problem. Consequently, it may be a little difficult to have the trust and confidence in the obtained recommendations. To address this issue, the authors have suggested, based on the work in the social sciences, that decision support systems provide modelling capability that will permit the triangulation of results. This, of course, entails that developers of DSSs need to consider alternative methods for analysing a given problem. This might be difficult to implement in a number of problem domains, but the added benefit of increased confidence and consequent greater usage of managerial aids would repay the initial investment.

Acknowledgements

The authors gratefully acknowledge the excellent programming assistance of Fan-Lau and Kuang-Ting Liu. This work was supported in part by a Dean's Summer Research Fellowship from the Ohio State University's College of Business to Dr. Balakrishnan.

References

[1] P.V. Balakrishnan and V.S. Jacob, 1992, A Genetic Algorithm for Product Design, paper presented at the ORSA/TIMS Conference, San Francisco, CA.
[2] R.W. Blanning, 1993, Model Management Issues and Directions, Decision Support Systems, Vol. 9, No. 1, pp. 9–18.
[3] D.T. Campbell and D.W. Fiske, 1959, Convergent and Discriminant Validity by Multitrait-Multimethod Matrix, Psychological Bulletin, Vol. 56, No. 2, pp. 81–105.
[4] A.M. Chang, C.W. Holsapple and A.B. Whinston, 1993,

Model Management Issues and Directions, Decision Support Systems, Vol. 9, No. 1, pp. 19–38.
[5] J. Couillard, 1993, A Decision Support System for Vehicle Fleet Planning, Decision Support Systems, Vol. 9, No. 2 pp 149–159.
[6] N. Denzin, 1970, The Research Act, Aldine: Chicago.
[7] B., Gavish, D. Horsky, and K. Srikanth, 1983, An Approach to Optimal Positioning of a New Product, Management Science, Vol. 29, No. 11, pp. 1277–1297.
[8] D.E. Goldberg, 1989, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley.
[9] P.E. Green, J.D. Carrol, S.M. Goldberg, 1981, A General Approach to Product Design Optimization via Conjoint Analysis, Journal of Marketing, Vol. 45, pp. 17–37.
[10] P.E. Green and A.M. Krieger, 1985 Models and Heuristics for Product Line Selection, Marketing Science, Vol. 4, No. 1, pp. 1–19.
[11] P.E. Green and A.M. Krieger, 1989 Recent Contribution to Optimal Product Positioning and Buyer Segmentation, European Journal of Operations Research, Vol. 41, 127–141.
[12] P.E. Green and A.M. Krieger, 1992, An Application of a Product Positioning Model to Pharmaceutical Products, Marketing Science, Vol. 11 (Spring), 117–132.
[13] P.E. Green and V. Srinivasan, 1978, Conjoint Analysis in Consumer Research: Issues and Outlook, Journal of Consumer Research, Vol. 5, pp. 103–123.
[14] S. Gupta and R. Kohli, 1990, Designing Products and Services for Consumer Welfare: Theoretical and Empirical Issues, Marketing Science, Vol. 9, No. 3, pp. 230–246.
[15] J.R. Hauser and P. Simmie, 1981, Profit Maximizing Perceptual Positions: An Integrated Theory for the Selection of Product Features and Price, Management Science, Vol. 27, No. 1, pp. 33–56.
[16] J.H. Holland, 1975, Adaptation in Natural and Artificial Systems, The University of Michigan Press, MI.
[17] R. Kohli and R. Krishnamurti, 1987, A Heuristic Approach to Product Design, Management Science, Vol. 33, No. 12, pp. 1523–1533.
[18] R. Kohli and R. Sukumar, 1990, Heuristics for Product line Design using Conjoint Analysis, Management Science, Vol. 36 pp. 311–322
[19] A.D. Shocker and V. Srinivasan, 1974, A Consumer-Based Methodology for the Identification of New Product Ideas, Management Science, Vol. 20, pp. 921–937.
[20] F. Zufryden, A Conjoint-Measurement-Based Approach for Optimal New Product Design and Product Positioning, in Analytical Approaches to Product and Market Planning, Ed. A.D. Shocker, Marketing Science Institute, MA.

P.V. (Sundar) Balakrishnan obtained his Ph.D. from the Marketing Department of the Wharton School of the University of Pennsylvania in 1988. He received his A.M. from the same place in 1987, an M.S. from the University of Texas at Arlington in 1983, and a B.Tech. from I.I.T., Delhi in 1981. He is currently an Assistant Professor in the Bothell Business Administration Program of the University of Washington. At the time of this writing he was on the Marketing Faculty of the Ohio State University. The specific substantive areas of his research interest are directed toward studying Channel/Sales Negotiations and Product Management with a focus on the distinctive approach of employing an interdisciplinary perspective by integrating behavioral concepts with analytical rigor. From a methodological standpoint he is interested in mathematical modelling of marketing phenomena and newer marketing research methodologies. His research has either been published or accepted for publication in journals such as **Management Science, Journal of Consumer Research, Organizational Behaviour and Human Decision Processes, Environment and Planning B, European Journal of Operational Research, Psychometrika.**

Varghese S. Jacob obtained his Ph.D. degree in Management, majoring in Management Information Systems, from Purdue University. He is currently Associate Professor of Management Information Systems and Associate Director of the Centre for Information Technologies in Management in the Max M. Fisher College of Business at The Ohio State University. His research interests are in machine learning, and design of intelligent systems, distributed systems, decision support systems, and group and organizational decision support systems. His publications include articles in **Decision Support Systems, IEEE Transactions on Systems, Man, and Cybernetics, Information Systems Research, Psychometrika**, and **International Journal of Man-Machine Studies**. He serves as an Associate Editor for **Decision Support Systems** and an ad hoc reviewer for several academic journals. He is a member of The Institute of Management Science (TIMS), IEEE Computer Society, Association of Computing Machinery (ACM) and the American Association of Artificial Intelligence (AAAI).