Genetic Algorithms for Product Design

P. V. (Sundar) Balakrishnan • Varghese S. Jacob
Business Administration Program, University of Washington, Bothell, Washington 98021
Department of Accounting and MIS, Max M. Fisher College of Business, The Ohio State University, Columbus, Ohio 43210

Product design is increasingly recognized as a critical activity that has a significant impact on the performance of firms. Consequently, when firms undertake a new (existing) product design (redesign) activity, it is important to employ techniques that will generate optimal solutions. As optimal product design using conjoint analysis data is an NP-hard problem, heuristic techniques for its solution have been proposed. This research proposes the use of and evaluates the performance of Genetic Algorithms (GA), which is based on the principles of natural selection, as an alternative procedure for generating "good" (i.e., close to optimal) solutions for the product design problem. The paper focuses on (1) how GA can be applied to the product design problems, (2) determining the comparative performance of GA vis-à-vis the dynamic programming (DP) heuristic (Kohli and Krishnamurti 1987, 1989) in generating solutions to the product design problems, (3) the sensitivity of the GA solutions to variations in parameter choices, and (4) generalizing the results of the dynamic programming heuristic to product designs involving attributes with varying number of levels and studying the impact of alternate attribute sequencing rules.

(Marketing; Product Policy; Conjoint Analysis; Heuristics; Buyers' Welfare)

1. Introduction

Optimal product design is recognized as a necessary and critical activity for firms seeking to survive in a competitive environment. It has increasingly received the attention of practitioners and academics due to very high rates of failure of new products and their associated losses (Business Week 1993). The product design problem has been determined to be NP-Hard (Kohli and Krishnamurti 1989) and as such heuristic solutions to the problem have been proposed by several researchers in the past (see Green and Krieger 1989 for a review). In this paper, we propose the use of an adaptive search technique discussed in the machine learning literature, namely Genetic Algorithms (GA), to obtain solutions to product design problems. Furthermore, we study the performance of the dynamic programming heuristic of Kohli and Krishnamurti (1987) to product designs involving attributes with varying number of levels and alternate attribute sequencing rules. The results obtained from both techniques are also compared.

The GA approach, first proposed by Holland (1975), seeks to mimic the approach of nature in the evolution of species; i.e., is based on the principles of population genetics to guide the search which results in the "survival of the fittest". This approach requires the specification of the candidate solutions in a binary or nonbinary string format. These strings are analogous to chromosomes in nature. A chromosome (product profile) in turn is composed of genes (attributes) each of which can take on a number of values (attribute levels) called alleles. Thus, for instance, we could have the hair color *gene* at a specific position or *locus* in the string, which can take on a specific *allele* value black, brown, or red. In each generation (iteration) the selection of candidate solutions to participate in the creation of offspring (new candidates) is based on their ability to survive in the competitive environment (i.e., its fitness). The GA approach has defined operators to perform activities corresponding to the creation of new chromosomes as a result of

0025-1909/96/4208/1105$01.25

mating and chromosome modification as a result of mutation.

A candidate solution string for product design is defined as per the specification within the conjoint or hybrid conjoint analysis framework and contains specific information about the product characteristics. A key feature of the proposed approach is that it is flexible enough to accommodate alternate fitness criteria in evaluating the candidate solutions ranging from the traditional deterministic rules such as share-of-choices to complex, composite choice rules. In §3, we illustrate the mapping of the product design problem to the GA structure and provide a more detailed description of the genetic algorithms process and its basic parameters.

The results across 192 data sets, in our study, indicate that the GA approach provides solutions which are 99.13% and 99.92% of the optimal product profile when the objectives of the problem are maximizing the share-of-choices and the buyers' welfare, respectively. This result is a statistically significant improvement of about 3% and 1% over the solutions identified by the dynamic programming heuristic of Kohli and Krishnamurti (1987, 1989) for the two objectives. Although the GA approach is an iterative procedure, the solutions even for the largest problem in the set were obtained within a minute.[1]

The paper is organized as follows; in §2 we briefly describe the product design problems, the difficulties with obtaining an optimal solution, the different solution techniques used to solve the problems and their associated shortcomings. In §3, we describe the genetic algorithm developed to address the above problems. In §4, the experimental methodology employed and the associated performance results of the proposed algorithm are presented, and in §5 the conclusions and future directions are discussed.

2. The Product Design Problem

The associated negative impact on organizations due to the failure of new products makes it imperative to design optimal products prior to their launch in an increasingly competitive global market. 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 a variety of procedures ranging from self-explication to 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, in designing a new bar soap a product manager would be interested in determining for the attributes color and shape, the specific level of each, blue, green, or white (for color), and, oval, round, rectangular, and spherical (for shape) to employ in the proposed new product.

In this paper, we formulate the problem within the conjoint or hybrid conjoint analysis framework which has enjoyed considerable commercial success (Green and Srinivasan 1990). Within the framework of conjoint or hybrid 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 (Green and Srinivasan 1978, 1990; Green 1984). The end result of the data collection and analysis is a consumers part-worths' matrix. This matrix contains, for the representative sample, for each individual consumer the utility associated with each level of each attribute. 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. The criteria specified 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. Now a complete enumeration procedure to identify a single product profile that results in the highest share-of-choices may be undertaken (Green et al. 1981, Green and Krieger 1992). However, in realistic applications, as the number of attributes and attribute levels increases, the number of possible product profiles increases dramatically, and consequently it becomes infeasible even with high speed computers to obtain an optimal solution in a reasonable amount of time. This major limitation has, therefore, led to the specification of heuristics for solving the problem (Gavish et al. 1983,

[1] The current implementation uses Pascal and is on a Sun Sparc Station.

Hauser and Simmie 1981, Green et al. 1981). These approaches are, however, not without their limitations as illustrated by Kohli and Krishnamurti (1987). Furthermore, other researchers such as Dobson and Kalish (1988, 1993), Green and Krieger (1985), and McBride and Zufryden (1988) have examined an alternative approach of selecting a new product from a restricted candidate set of products.

2.1. Problem Description

The typical mathematical formulation of the product design problem (Zufryden 1977, Kohli and Krishnamurti 1987) considers that there are a total of K relevant attributes in the product category. Further each attribute k $(=1, \cdots K)$ has L_k levels, and, there are N consumers in the sample. Based on the consumer utilities (part-worths) data then, the product design problem becomes one of selecting a specific level for each of the K attributes in order to satisfy the manufacturer's objective. In addition to the matrix of consumer part-worths, the product manager typically also has information on the current (or projected) competitive environment. In other words, the manager must be aware of the competitive product offerings in order to design a product that will survive and effectively compete in this market place. The managerial objective, thus, also 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 (Gupta and Kohli 1990). Finding that product profile that results in the largest overall consumer utility is referred to as the buyer's welfare problem.

2.2. Dynamic Programming Heuristic

Kohli and Krishnamurti (1987) have used a dynamic-programming heuristic to solve both the share-of-choices problem as well as the buyers' welfare problem. This heuristic has been shown to work well in that it has outperformed randomly generated products, the product profiles identified by the greedy heuristic and the lagrangian-relaxation heuristic (Kohli and Krishnamurti 1987, 1989; Kohli and Sukumar 1990). The dynamic-programming 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 combination is determined by the appropriate choice of evaluation functions, i.e. share-of-choices or buyers welfare.

There are some limitations with this approach. First, at each stage one loses information since only the best partial profiles are kept. This process could eliminate, at even a very early stage, from further consideration the partial profile of the optimal solution. Second, as discussed in Kohli and Krishnamurti (1989), the heuristic is sensitive to the order in which the attributes are considered. Third, to consider interactions, the approach requires *a priori* identification of the interactions and then sequencing the attributes successively (Kohli and Krishnamurti 1989). This limits the intermediate-stages attributes to participating in two two-way interactions (with the preceding and succeeding attributes). More complex interactions can be handled by creating composite attributes, i.e., combine attributes i and j; which, however, results in the composite attribute having as many levels as the product of the number of levels $(L_i * L_j)$ in the two original attributes. The presence of interactions, unfortunately, limits the number of possible random orderings of the attributes and thereby limiting multiple runs that help identify better solutions. In the next section, we describe and illustrate the use of a *maximally different* method to get a better fix on the answer (Balakrishnan and Jacob 1995; Campbell and Fiske

1959) that ameliorates some of the above problems while providing a flexible framework to more easily incorporate alternative managerial criteria.

3. Genetic Algorithms

The concept of Genetic Algorithms (GA) was first proposed by Holland (1975). 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.

3.1. Overview of Algorithm Process

The basic GA process can be visualized as follows (Michalewicz 1992, Grefenstette 1986):

Step 1: $t \leftarrow 0$.
Step 2: Generate initial population, POP(t) either randomly or using a heuristic.
Step 3: Evaluate each of the strings in POP(t).
Step 4: IF stopping condition satisfied THEN Stop ELSE continue.
Step 5: $t \leftarrow t + 1$.
Step 6: Select strings from POP($t - 1$) to create POP(t).
Step 7: Apply genetic operators on strings in POP(t).
Step 8: Evaluate each of the strings in POP(t).
Step 9: GO TO 4.

The candidate solution set of strings (i.e., product profiles) generated in Step 2 above forms the initial chromosome pool (i.e., initial generation). The size of the chromosome pool (i.e., the number of strings) M, is generally maintained in successive generations. The genetic operators used to generate candidate products are:

1) *Reproduction*: A subset of the products m ($<M$) from the population of size M is selected based on their fitness and copies of their profiles are generated;

2) *Crossover*: Pairs of reproduced product profiles are chosen and along specific positions on the strings genetic material between the two strings are exchanged leading to offspring (i.e., two new product profiles);

3) *Mutation*: During this process, a product profile is randomly chosen from the population and the value at a specific location (attribute level) in the string is modified.

The technique basically exploits the fact that a "good" product profile contains some features (substrings) that are desirable and hence contribute to its being evaluated highly. When one exchanges genetic material between two good strings the expectation is that it will frequently produce offspring that combines the good features of the parents. Thus, as the iterative process continues, i.e., one moves from one generation to the next, the product profiles making up the subsequent population are going to be "fitter" than those of the preceding generation.

The GA approach has several advantages in the manner in which it performs its search process. This also distinguishes GA from the typical optimization and search procedures (Goldberg 1989). A key feature of GA is that the search is conducted from a population of points rather than a single point thus increasing the exploratory capability of GA. In contrast with other gradient search techniques which need additional knowledge such as the differentiability of the function, GA use the objective function or payoff directly, in that the candidate product profiles are evaluated based on the specified objective. Another feature of GA is that it works with a direct coding of the parameter set rather than the parameters themselves. This implies that optimization problem domains that are characterized as discontinuous, high-dimensional, and multimodal (as in this case) are especially suited for GA as opposed to gradient or random search techniques. Additionally, GA work by evaluating completely specified candidate solutions, unlike the DP heuristic which designs products by building profiles sequentially one attribute at a time. If each string is of length L, then GA adds to its store of information on the performance of 2^L hyperplanes (for binary strings) represented by the structure of the particular string. Finally, another feature of GA is that they lend themselves naturally to implementation in parallel processing environments leading to the ability to exploit newer technologies in this domain resulting in even faster computational times.

GA have been successfully used in a wide variety of applications. For example, Goldberg (1989) has used GA for optimization of pipeline systems; Cleveland and Smith (1989) for scheduling flow shop releases; Liepins and Potter (1991) use GA for multiple fault diagnosis; Holsapple et al. for designing adaptive decision support

systems and flexible manufacturing systems scheduling (Holsapple et al. 1993a,b). An extensive discussion on GA and their applications can be found in Goldberg (1989), Davis (1991) and Koza (1991).

3.2. Application of GA to Product Design

For a problem to be solved within the GA paradigm, one needs to formulate the problem using a string structure. In the product design problem, we consider a chromosome (i.e., string) to represent a product. The attributes of the product would correspond to genes and the levels, i.e., values the attribute could take, to alleles.

String Specification. We employ the standard dummy variable coding procedure (i.e., binary string) representation of the product to readily make the connection to both the conjoint and GA literature. The binary string implementation of a product is done as follows: as noted before, if there are K attributes in the product category and each attribute j $(=1, \ldots, K)$ has L_j levels, then a string will be defined by 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 substrings (i.e., attributes) where the length of a substring j would be $L_j - 1$. Thus, for example, if we had a product category soap, represented by three attributes A1 (shape), A2 (color), and A3 (scent), and attribute A1 has three levels (rectangle, square, spherical), attribute A2 four levels (red, green, yellow, white) and attribute A3 three levels (fruity, flower, antiseptic), then an instance of a product string would be, 01 100 01, which corresponds to a spherical shaped soap with green color and an antiseptic scent. Thus, a one in a specific attribute level denotes the presence of that level and a zero the absence of the level. Also, all part-worths are measured relative to the dummy level. This representation simplifies the evaluation of the product profile as well as provides a natural structure for considering interactions as well as technological infeasibility.

Once the product category in terms of the number of attributes and levels is defined, an initial population is generated. In other words, M strings (i.e., candidate products within the category) to make up a population are generated either randomly or by some heuristic procedure.[2] In general, the initial population should contain as diverse a representation as possible since diversity improves the sampling of the search space. The population is maintained in a matrix $\mathbf{POP}_{M*P}$.

Evaluation. The solution process here is an iterative procedure. The system first evaluates each product profile in the initial population. The evaluation is done by using the part-worth utilities which are obtained from conjoint analysis and which are stored in matrix **BETA** of size $N \times P$, where N is the number of consumers considered and P is, as defined earlier, the length of the string (assuming dummy variable coding). The evaluation is done as follows:

1. Determine for each of the M products the utility associated with it by each customer from the **BETA** (part-worths) matrix, so if **PRODUTIL** (of size $N*M$) is the matrix in which the resulting information is stored, then $\mathbf{PRODUTIL} = \mathbf{BETA}*\mathbf{POP}^T$. Technically, as the population matrix is a sparse matrix we need to perform only $N*K*M$ additions.

2. *Share of Choices.* For each product m $(=1 \cdots M)$ 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**, which contains the utility associated by each consumer for their status quo product. If the utility associated with the new product m is greater than that of the current favorite product (by a idiosyncratic threshold value) for a consumer, then the new product would be chosen. By counting the number of consumers who would choose the new product over the competition we have a measure of the potential number of consumers. This count is used to specify the fitness of a specific product (string).

Buyers' Welfare. In the buyers' welfare problem, the utilities in each column of **PRODUTIL** are added, and this provides an aggregate measure of the social welfare associated with a product. The fitness associated with a candidate product profile in the buyer's welfare case is this aggregate utility value.

Other evaluation criteria such as sellers return, Nash welfare function, and nondeterministic choice rules to specify the consumer behavior can be easily implemented. 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.

[2] The initial population for the current study was generated randomly.

3.3. GA Parameters

We now provide a brief overview of the genetic algorithm parameters employed in the current study that are particularly germane to the product design problem. We begin by discussing the operationalization of the GA operators to the specific class of string structures defined above.

Reproduction. Although one can choose the strings for reproduction in many different ways, the current implementation selects the "s" ($=M/2$) fittest strings from the population to survive intact into the next generation. This is referred to as an *elitist strategy* and has been shown to work well in several different problems by Grefenstette (1986). This elitism ensures that the better candidate profiles are preserved and thus also have an opportunity to produce offspring unlike in the case of probabilistic selection where they could be deleted from further consideration.

Crossover. In crossover, randomly picked pairs of strings from the set of reproduced strings exchange genetic material to produce offspring. Unlike the other problems typically examined in the GA literature, in the case of the product design problem the unit of interest is the attribute which has been coded as a substring,[3] hence, crossover is done at the substring level. Thus, suppose we have two feasible strings represented as: s_1 = 10 001 00 and s_2 = 01 100 01.

Now consider the simple *single-point* crossover where all information from strings s_1 and s_2 from beyond an arbitrary point to the end of the chromosome are swapped. For instance, if we arbitrarily select the crossover point to be at the end of substring 1, i.e., attribute 1, then substrings representing attributes 2 and 3 are swapped between the two strings, resulting in two new strings: s_3 = 10 100 01 and s_4 = 01 001 00.

The exchange of attributes can be implemented in a number of ways. The *uniform crossover* has, however, been shown to out perform the traditional approaches such as the single-point crossover (Syswerda 1989). In the approach that we take (i.e., uniform crossover with 0.5 probability) the attributes that are exchanged are uniformly distributed throughout the chromosome rather than having contiguous segments of attributes exchanged. As the number of attributes considered in the product design problems are relatively not very large we implement a deterministic version of the uniform crossover rule in that exactly half of the attributes, selected randomly, participate in each crossover. The impact of varying this probability is examined empirically in §4.

Mutation. Once a new set of profiles is created by the use of the above two operators, from amongst these members we provide each string a chance to mutate. The mutation parameter is considered to be a secondary search operator which aids in exploring a larger area of the search space by increasing the variability of the population (Grefenstette 1986). The number of mutations in a generation has to be balanced between the desire to keep it high in order to explore larger areas of the search space and kept very small in order to prevent an essentially random search.

For our problems, 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. Thus, if string s_4 shown above, is mutated by changing the level of the second (color) attribute from white to yellow, it results in an entirely new candidate string, 01 010 00. In the typical GA approach, every bit position of every string has some specified probability (i.e., the "traditional" mutation rate) of undergoing a random change. This means that for the expected number of mutations per generation in our approach to be comparable with the mutation rate values typically employed (e.g., 0.001 by Syswerda) we have to set our mutation rate equal to the product of the traditional mutation rate value and the length of the string.

The process of reproduction and crossover generates additional strings. Hence if one starts with an initial population of M strings, the number of strings in the population keeps on growing as reproduction and crossover are performed. In the current implementation, the process

[3] We have chosen to use the binary (dummy-variable) format to readily make connection to both the GA and conjoint literature as well as for ease of computation. It should, however, be noted that in the traditional implementation of GA a bit is considered as an attribute, hence, crossover occurs at the bit level (or at each position). In our context, we could also code the string in a nonbinary format, for example, the string 10 001 00 could be coded as 2 4 1, representing level 2 of attribute 1, level 4 of attribute 2 and level 1 of attribute 3. In this case crossover can be done at each position of the string.

continues till the size of the population becomes $2M$ at which point, M strings having the lowest fitness are culled, bringing the population back to size M. From this new population again strings are chosen for reproduction and the process continues till a stopping criteria is met. For instance, if $M = 100$, then copies of the 50 best strings are made. Now, 25 pairs of these best 50 are randomly selected (with replacement) to mate leading to 50 offspring. Now the mutation operator is applied to the 50 copies and the 50 offspring leading to 100 new profiles. This results in a total of 200 strings (new and old profiles) which are then culled down to 100 candidate profiles which then form the basis of the next iteration.

Stopping Rule. The process of creating new generations and evaluating them continues till a stopping condition is met. Either a prespecified fixed number of iterations, or a minimal percentage change in the average fitness of a number of best strings over a few previous generations can be used as the stopping criteria. Using purely the number of iterations as a stopping criteria can turn out to be either wasteful or inadequate. If the process converges quickly to a solution then iterations will be performed with no improvement in the solution. On the other hand if few iterations are specified then the process may generate solutions that could be improved substantially if it were provided an opportunity to continue. The moving average rule is used in our study as the stopping criteria as it provides a good indication of convergence to a solution.

The time complexity of the algorithm can be seen to be of the order of $O(IKMN)$ for the GA, where I is the number of iterations, K the number of attributes, M the population size and N the number of consumers whose utilities are used for evaluation. In our current implementation the upper bound on the number of iterations has been set to 100 when the degree of improvement rule is used.[4] This is somewhat larger than the case for the single run of the heuristic DP which has an order of complexity, $O(\Sigma_{j=2}^{K} NL_jL_{j-1})$. The storage requirements of the GA approach requires the maintenance of the utility of N consumers, i.e., $(N \times P)$, where P is as defined before, the current population which is $M \times P$ which grows to $(2M)P$ before it is brought down to $M \times P$, the status quo products utility for the N consumers, $N \times 1$ (note that this is not required when using the relative BETA approach of Kohli and Krishnamurti (1989)) and the storage of the current value of the average fitness of the previous generations. Thus, the maximum storage requirement for the algorithm is $P(2M + N) + (N + 1)$. Further, if we keep the S best strings from each generation then for I iterations we have $P(2M + N + IS) + (N + 1)$ as the storage requirements. It can be shown easily that the worst case performance for the algorithm is arbitrarily bad.[5]

When the 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 (this would, however, increase the storage requirements). 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. Performance Evaluation

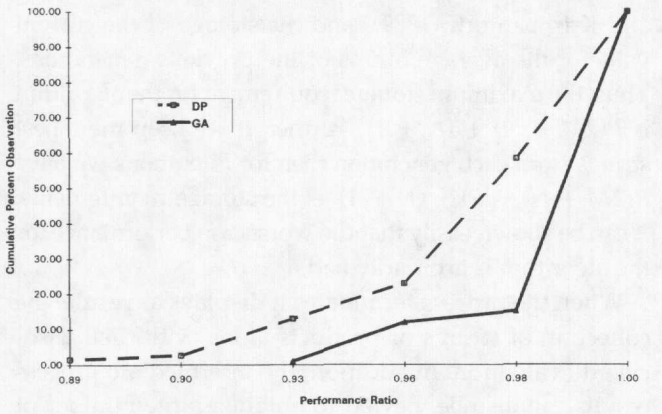
In this section, we describe the methodology employed to evaluate the performance of the proposed heuristic. In particular, we compare the performance of the proposed genetic algorithm with the dynamic programming heuristic and also with the complete enumeration procedure where feasible. The data for the problems examined are generated using a procedure described in Kohli and Krishnamurti (1987).

The methodology is described in three parts. First, we provide the results of comparative performance of the genetic algorithm and the DP heuristic relative to the complete enumeration solutions in 192 data sets for

[4] Note that in all cases the solution was obtained well before the maximum limit and typically under 15 iterations.

[5] Note that if the randomly generated initial population consisted of M identical strings which was not preferred by any of the consumers over their status-quo, then crossover would not create any new strings. Additionally, as mutation is a probabilistic mechanism, there is a possibility that no mutations occur resulting in the next generation also consisting of M identical copies of the nonpreferred product. If this process repeats itself over the next X generations, at which point the stopping condition is met, the solution obtained is a product not selected by any of the consumers. However, as illustrated by the results the performance is substantially better than the worst case performance.

Figure 1 **Share of Choices—Cumulative Distribution of Performance (Relative to Optimal) Ratio Across 192 Simulated Problems for GA and Heuristic DP**



both the share-of-choices and the buyers' welfare problems. Next, we describe and present the results of our computational experience with the genetic algorithms for sensitivity to changes in parameter values. Finally, we examine the relative performance of these two heuristics on larger data sets.

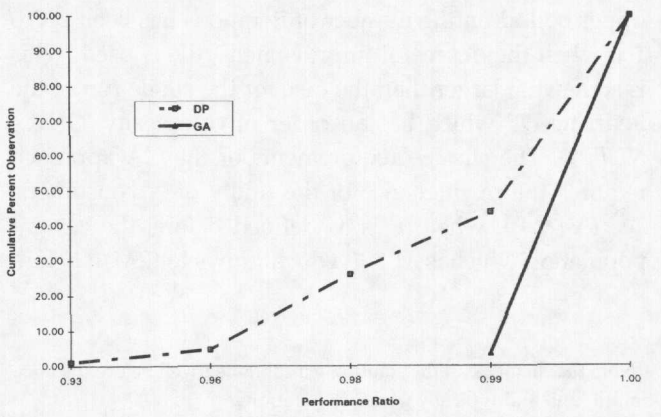
4.1. Comparative Performance Relative to Optimal

Forty eight problems were generated using a $(3 \times 4 \times 4)$ full factorial experimental design which varied by the number of attributes in the product category (4, 6, 8), number of levels in each attribute (2, 3, 4, 5) and number of consumers in sample (100, 200, 300, 400). In addition, four replications of each problem were generated leading to a total of 192 data sets that were investigated. The part-worths generated randomly from a uniform distribution are normalized within respondent. In addition, each individual was assigned a idiosyncratic status-quo product profile that was randomly generated.

Each data set was then analyzed to identify product profiles that solve the objective functions of maximizing the share-of-choices and the buyers' welfare. The profiles and the resulting evaluations (i.e., share-of-choices, buyers' welfare) obtained by the genetic algorithm and dynamic programming heuristics are compared with the optimal solution found by a complete enumeration procedure. For GA, in addition to the parameters described earlier, the population size, M, was set to 100, the mutation rate was set to 0.3, and the process was terminated if the moving average over five previous generations was less than 2% of the fitness of the three best strings of the current generation.

Following Kohli and Krishnamurti (1989), Figures 1 and 2 show the cumulative fraction of problems with a performance ratio less than a specified value for GA and DP for the share-of-choices and buyers' welfare problems. In both cases the GA solution dominates the heuristic DP solution. On average, the share-of-choices of the best product-profile identified by the genetic algorithm across 192 cases was 99.13% of the share-of-choices of the optimal-product-profile. This compares favorably with the overall average of 96.67% of the optimal share-of-choices for the dynamic-programming heuristic. In addition, the standard deviation is tighter for the genetic algorithm (of 0.016) as compared to that (of 0.031) for the dynamic-programming heuristic. Similarly, for the buyers' welfare problem, the genetic-algorithm identifies product profiles which are on an average 99.92% of the welfare provided by the optimal-product-profile as compared with an average of 98.76% of the optimal for the dynamic-programming heuristic. The margin of improvement in performance for the genetic algorithm over the dynamic-programming heuristic is relatively less for the buyers' problem in part due to the ceiling effect, however, the standard deviation is considerably tighter (0.0028 for GA as opposed to 0.0165 for DP). Finally, the hit rate (i.e., the number of cases in which the optimal solution was identified) for GA and the dynamic-programming heuristic was 123 cases and

Figure 2 **Buyers Welfare—Cumulative Distribution of Performance (Relative to Optimal) Ratio Across 192 Simulated Problems for GA and Heuristic DP**



51 cases out of a total of 192 cases, respectively for the share-of choices problem and 175 and 82 cases, respectively for the buyers' welfare problem.

4.2. Sensitivity Analysis of Genetic Algorithm

The sensitivity of the GA performance to changes in the parameters' values was also investigated. A $5 \times 4 \times 3 \times 2$ full factorial experimental design was employed to assess the impact of different Mutation rates (0, 0.01, 0.1, 0.25, 0.3), number of attributes participating in the Crossover (0, $K/4$, $K/2$, $3K/4$), chromosome pool Population Sizes (50, 100, 200), and the Stopping rules' degree of improvement (2%, 0.2%). Thus, 120 separate runs of GA were conducted on one of the replicates of the data sets (with 400 consumers, 8 attributes each with 5 levels) for both the share-of-choices and the buyers' welfare problems.[6] The results of the sensitivity analysis runs were then subjected to a series of analysis of variance (ANOVAs).

For the share-of-choices problem the product profile identified by the GA across the 120 runs, on average, exceeds 96.8% of the share of the optimal profile. Also, on average, it takes approximately 7.35 iterations for the GA to achieve convergence and is accomplished in less than 14 seconds of CPU time. Consequently, the GA is able to find near optimal solutions by evaluating fewer than one-fourth of one percent of the total number of possible product profiles to choose from, which for these problems with eight attributes each with five levels exceeds a third of a million (390,625).

The ANOVA of the sensitivity analysis for the share-of-choices problem with a main-effects model has an R^2 of 0.504 and is statistically significant ($p < 0.05$). The results indicate that the main effects of Population Size, Crossover, and, Stopping Percentage significantly affect the quality of the GA solution. The impact of changes in the significant GA parameters are all in the expected direction. Specifically, as the size of the population increases the fitness (share-of-choices) of the identified profile, as a percent-of-the-share-of-choices of the optimal profile, also increases. Also, as the Crossover mechanism is disabled (i.e., set to 0) the quality of the obtained product strings significantly decreases. Additionally, as the stopping parameter for convergence is tightened from 2% to 0.2% the quality of the obtained solution goes up as expected. Mutation, on the other hand, had no significant main effect ($p = 0.175$). However, the best performance was at the highest mutation rate, directionally continuing to suggest evidence of its role as a secondary search operator to explore further spaces.

The impact of the four main-effects model on CPU times was also significant ($p < 0.05$) with R^2 of 0.798. The main effects of Population Size, Crossover, and Stopping Percentage significantly affected ($p < 0.05$) the CPU time in the expected direction with Mutation having no impact.

Similar analyses on 120 simulation runs was carried out for the buyers' welfare problem resulting in an R^2 of 0.724 ($p < 0.05$). All of the results were similar to that of the share-of-choices problem. The buyers' welfare of the product profile identified by the GA across the 120 runs, on average, is 97.9% of the optimal profile. Also, on average, it takes approximately 8.48 iterations for the GA to achieve convergence which is accomplished in less than 14.73 of CPU time. The impact of the main effects model on CPU times was significant with R^2 of 0.876. Consistently, the Population Size, Crossover, and Stopping Percentage main effects significantly affected the CPU time in the expected direction with Mutation having no impact.[7]

4.3. Impact of Varying Attributes' Size

In this study, we investigate the performance of two heuristics on larger data sets in which the attributes have a varying number of levels. In addition, we examine the impact of different sequencings of attributes on the performance of the DP heuristic and compare it with a single run of the GA.

[6] In a separate experiment on another data set, it was determined that, for both the share-of-choices and buyers' welfare problems, the modification in the stopping rule of increasing the number of strings from 3 to 20 in computing the average population fitness, had no impact on the quality of the obtained solutions ($p > 0.8$). It did, of course impact significantly the number of iterations and CPU time it took for the GA to converge. Consequently, we did not consider this parameter any further.

[7] A similar analysis was conducted on another replicate of a data set of the same size leading to similar findings. More detailed information is presented in a technical appendix that can be obtained from the INFORMS office.

Table 1 ANOVA of Simulation Runs on Data Set with Eight Attributes at Five Levels Each, and 400 Consumers

			Share of Choices				Buyers' Welfare			
Parameters	Level	No. of Runs	Mean Evaluation	*p*-Value	Mean CPU Time	*p*-Value	Mean Evaluation	*p*-Value	Mean CPU Time	*p*-Value
POPSIZE				*0.0001*		*0.0001*		*0.0001*		*0.0001*
	50	40	0.95895		8.258		0.96882		9.016	
	100	40	0.96903		11.726		0.98041		13.333	
	200	40	0.97644		20.800		0.98856		21.825	
MUTATN				*0.1753*		*0.1791*		*0.5961*		*0.536*
	0	24	0.966735		12.710		0.97689		14.358	
	0.01	24	0.964677		13.566		0.97744		14.303	
	0.10	24	0.969136		13.261		0.97951		14.756	
	0.25	24	0.96708		13.533		0.98166		14.809	
	0.30	24	0.97308		14.904		0.98083		15.394	
XOVER				*0.0001*		*0.0069*		*0.0001*		*0.0001*
	0	30	0.954047		12.246		0.95042		12.474	
	2	30	0.972016		14.110		0.98793		15.039	
	4	30	0.977229		14.943		0.98965		15.732	
	6	30	0.969273		13.080		0.98906		15.650	
STPGPRCNT				*0.0489*		*0.0001*		*0.0366*		*0.0001*
	2.0%	60	0.965912		11.208		0.97691		12.026	
	0.2%	60	0.970370		15.981		0.98162		17.422	
Overall Mean		120	0.96814		13.595		0.97927		14.724	
R^2			0.504	*0.0001*	0.798	*0.0001*	0.724	*0.0001*	0.876	*0.0001*
No. of Iterations			7.35				8.475			

Toward this end, a simulation study was conducted on two problem sizes, each comprised of 200 consumers and 9 attributes. The relatively smaller problem was roughly the same size as the largest problem, in terms of the number of possible product profiles, in the Kohli and Krishnamurti study (1987). In the first (second) problem set, there were more (fewer) attributes with a relatively large (>5) number of levels than attributes with a small (≤3) number of levels. Ten replications of each of the two sizes of data were generated. The total number of possible product profiles in the larger data sets numbered 870,912 [$(9 \times 8 \times 8 \times 7 \times 6) \times (2 \times 2 \times 3 \times 3)$] and in the smaller data sets numbered 326,592 [$(9 \times 8 \times 7 \times 6) \times (2 \times 2 \times 3 \times 3 \times 3)$].

As both of these sets of problems are fairly large a complete enumeration procedure to identify the optimal solutions was not done. Instead, we focused on comparing the results of a single run of the GA against ten different runs of the DP heuristic. The difference in the DP heuristic runs was in the specific sequencings of the attributes. The order in which the different attributes (with varying levels) are considered is an important factor as it affects the number of different configurations that are being evaluated and retained at each stage.

Rather than merely perform ten random runs on each data set, we employ ten sequencing rules to obtain a better understanding of their impact on the performance of the DP heuristic. The ten rules employed in this study, which specify the stage at which each particular attribute is considered by the DP heuristic, are as follows: *Ascending order* (in which the attributes with the fewest number of levels are considered first), *Descending order* (the attributes with the largest number of levels are considered first), *High-Low* (of the remaining attributes an alternating arrangement of the attribute with the largest number of levels followed by the attribute with the fewest number of levels), *Mirror High-Low* (a mirror image of the previous asymptotic-sawtooth

pattern), *Low-High* (of the remaining attributes an alternating arrangement of the attribute with the fewest number of levels followed by the attribute with the largest number of levels), *Mirror Low-High* (a mirror image of the previous ordering), *U-shape* (the attribute with the largest number of levels first, the next largest number of levels last, and so on), *Bell-shape* (the attribute with the fewest number of levels first, the next fewest number of levels last, and so on), *Step* (the attribute with the largest number of levels first followed by the two attributes with the fewest number of levels, and then the two attributes with the largest number of levels of the remaining and so on), and *Mirror Step* (the reverse of the previous ordering).

The results of our current study indicate that the single run of the GA does better, worse, and ties the best DP heuristic solution in 11, 3, and 6 (out of 20) data sets, respectively for the share-of-choices problem. The single run of the GA does better, worse, and ties the best DP heuristic solution in 8, 3, and 9 (out of 20) data sets, respectively for the buyers' welfare problem. Furthermore, once the solution was determined for each of these sequencings, the ten rules were ranked in order of performance and an average ranking was calculated for each across all replications for the two sets of problems. This was done for both the share-of-choices and the buyers' welfare criteria. The Mirror Step attribute sequence, on average, gave the best results of the ten rules considered for both problems with the descending order and ascending order sequences also performing well. The bell shape, low-high, high-low and mirror high-low sequences consistently performed poorly.

Several factors play a role in the solution obtained by the DP. Based on the sequencing scheme a significant number of products are eliminated from consideration at each stage of the algorithm. Further, the sequencing also affects the number of partial product configurations considered at each stage. Indeed the effect of both these factors can be seen by considering the evaluation procedure of the heuristic. Since the primary focus of this the study was not on the attribute sequencing rules per se further study was not performed on them. Future research which takes into account such factors as the relative importance of an attribute, etc. are required to further study the DP heuristic behavior.

5. Conclusions

This paper proposed and demonstrated the use and advantages of Genetic Algorithms for solving product design problems. The GA approach has given us significantly superior results as compared to the heuristic DP results in the design of a single product for the data sets investigated. The GA procedure on average not only provided a better solution with a smaller standard deviation than the heuristic DP approach, of the 192 data sets investigated, it also obtained the optimal solution more often then the DP heuristic. The CPU times for the DP heuristic ranged from about 0.5 seconds for the smaller problem sets to over four seconds for the largest problems and for complete enumeration it varied from 0.5 seconds to about 10 hours.

A nice feature of the GA approach is its ability to consider both interaction between attributes and technological infeasibility within the same construct. Interaction between attributes essentially implies that a consumer would have a greater (or lesser) preference for a product which has say, the combination of high gas mileage and a V6 engine, than what the additive utility model of considering each item in isolation would suggest. Within the GA construct one would augment the **BETA** matrix to have columns that correspond to the utility estimated with the presence of selected interaction terms. As far as technological infeasibility is concerned the **BETA** matrix would be further augmented by a column which specifies a large negative value for the technological infeasible combination of attribute levels. For an alternative approach to the use of penalty functions within GA the methodology of Michalewicz and Janikow (1991) is also recommended. The GA approach is also flexible enough to permit one to take into account *idiosyncratic* interactions effects.

In addition to the advantages discussed above, the GA approach provides a technique which is founded on a concept that is significantly different than that of mathematical programming. Consequently, when used in conjunction with more traditional heuristics, this allows us to design decision aids such as decision support systems that provide the ability to triangulate on a solution. This approach is significant as often real life problems requiring DSS aids are complex and NP-hard and heuristic approaches used, by definition, do not guarantee optimal solutions. In such situations, when

one is left with the problem of justifying the solution or providing benchmarks, researchers are recommended to use *maximally different* methods in order to get a better fix on the answer (Campbell and Fiske 1959). 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 (Denzin 1970). Consequently, one would have greater confidence in the results if they were generated by alternate approaches. Since, as noted earlier, any one heuristic may fail in particular circumstances, a way of checking whether or not the result obtained with a specific heuristic can be considered "good" (close to optimal) is by bench marking against the result obtained using another maximally different approach (Balakrishnan and Jacob 1995).

The results of the study thus suggest that the GA approach seems to hold significant promise in generating better solutions to the product design problem than the currently used techniques. Additionally it is possible to use it in conjunction with other, more traditional, techniques to increase the confidence in the solutions generated. In the current study, the initial population of candidate profiles was randomly generated. However, one advantage of the GA approach is that the initial population can be populated with solutions obtained using other techniques; the algorithm then could potentially improve on the existing solution. For future research, this approach needs to be extended to accommodate product line decisions, household-level negotiated purchases, and consideration sets. We also encourage the evaluation of this fairly flexible approach to other similar large problems in marketing such as advertising, scheduling, and retail site locations.[8]

[8] This work was supported in part by a Dean's Summer Research Fellowship from the College of Business to Dr. Balakrishnan while he was on the Marketing Faculty at The Ohio State University. The authors gratefully acknowledge the excellent programming assistance of Fan-Lau, Kuang-Ting Liu, and Budi Yuwono; and the suggestions of Shobana Balakrishnan. They thank Profs. Kohli and Sukumar for providing the code for the simulation data generator. They also thank the Departmental Editor Josh Eliashberg, the Associate Editor, and the anonymous referees for their helpful comments.

References

Balakrishnan, P. V. and V. S. Jacob, "A Genetic Algorithm for Product Design," presented at The Institute of Management Sciences the Marketing Science Conference, London, UK, 1992.

—— and ——, "Triangulation in Decision Support Systems: Algorithms for Product Design," *Decision Support Systems*, 14 (1995), 313–327.

Business Week, "Flops: Too Many New Products Fail," August 16, 1993, 76–82.

Campbell, D. T. and D. W. Fiske, "Convergent and Discriminant Validity by Multitrait-Multimethod Matrix," *Psychological Bull.*, 56, 2 (1959), 81–105.

Cleveland, G. A. and S. F. Smith, "Using Genetic Algorithms to Schedule Flow Shop Releases," in *Proc. Third International Conf. on Genetic Algorithms*, J. D. Schaffer (Ed.), Morgan Kaufmann Publishers, San Mateo, CA, 1989, 160–169.

Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

Denzin, N., *The Research Act*, Aldine; Chicago, IL, 1970.

Dobson, G. and S. Kalish, "Positioning and Pricing of a Product-Line: Formulation and Heuristics," *Marketing Science*, 7, 2 (1988), 107–125.

—— and ——, "Heuristics for Pricing and Positioning a Product-Line Using Conjoint and Cost Data," *Management Sci.*, 39, 2 (1993), 160–175.

Eshelman, L. J., R. A. Caruana, and J. D. Schaffer, "Biases in the Crossover Landscape," *Proc. Third International Conf. on Genetic Algorithms*, J. D. Schaffer (Ed.), Morgan Kaufmann Publishers, San Mateo, CA, 1989, 10–19.

Gavish, B., D. Horsky, and K. Srikanth, "An Approach to Optimal Positioning of a New Product," *Management Sci.*, 29, 11 (1983), 1277–1297.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

Green, P. E., "Hybrid Models for Conjoint Analysis: An Expository Review," *J. Marketing Res.*, 21 (May 1984), 155–169.

——, J. D. Carrol, and S. M. Goldberg, "A General Approach to Product Design Optimization via Conjoint Analysis," *J. Marketing*, 45 (1981), 17–37.

—— and A. M. Krieger, "Models and Heuristics for Product Line Selection," *Marketing Sci.*, 4, 1 (1985), 1–19.

—— and ——, "Recent Contribution to Optimal Product Positioning and Buyer Segmentation," *European J. Oper. Res.*, 41 (1989), 127–141.

—— and ——, "An Application of a Product Positioning Model to Pharmaceutical Products," *Marketing Sci.*, 11 (Spring 1992), 117–132.

—— and V. Srinivasan, "Conjoint Analysis in Consumer Research: Issues and Outlook," *J. Consumer Res.*, 5 (1978), 103–123.

—— and —— "Conjoint Analysis in Marketing Research: New Developments and Directions," *J. Marketing*, 54 (October 1990), 3–19.

Grefenstette, J. J., "Optimization of Control Parameters for Genetic Algorithms," *IEEE Trans. Systems, Man, and Cybernetics*, SMC 16, 1 (1986), 122–128.

Gupta, S. and R. Kohli, "Designing Products and Services for Consumer Welfare: Theoretical and Empirical Issues," *Marketing Sci.*, 9, 3 (1990), 230–246.

Hauser, J. R. and P. Simmie, "Profit Maximizing Perceptual Positions: An Integrated Theory for the Selection of Product Features and Price," *Management Sci.*, 27, 1 (1981), 33–56.

Holland, J. H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

Holsapple, C., V. S. Jacob, R. Pakath, and J. Zaveri, "Learning by Problem Processors: Adaptive Decision Support Systems" *Decision Support Systems*, 10 (1993a), 85–108.

——, ——, ——, and ——, "A Genetics Based Hybrid Scheduler for Generating Static Schedules in Flexible Manufacturing Contexts" *IEEE Trans. Systems, Man and Cybernetics*, 23, 4 (1993b), 953–972.

Kohli, R. and R. Krishnamurti, "A Heuristic Approach to Product Design," *Management Sci.*, 33, 12 (1987), 1523–1533.

—— and ——, "Optimal Product Design Using Conjoint Analysis: Computational Complexity and Algorithms," *European J. Oper. Res.*, 40 (1989), 186–195.

—— and R. Sukumar, "Heuristics for Product Line Design Using Conjoint Analysis," *Management Sci.*, 36 (1990), 1464–1478.

Koza, J. R., *Genetic Programming*, MIT Press, Cambridge, MA, 1991.

Liepins, G. E. and W. D. Potter, "A Genetic Algorithm Approach to Multiple-Fault Diagnosis," in *Handbook of Genetic Algorithms*, L. Davis, (Ed.) Van Nostrand Reinhold, New York, 1991.

McBride, R. D. and F. S. Zufryden, "An Integer Programming Approach to Optimal Product-line Selection," *Marketing Sci.*, 7, 2 (1988), 126–140.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, 1992.

—— and C. Z. Janikow, "Handling Constraints in Genetic Algorithms," *Proc. Fourth International Conf. Genetic Algorithms*, R. K. Belew and L. B. Booker (Eds.), Morgan Kaufmann Publishers, San Mateo, CA, 1991, 151–157.

Shocker, A. D. and V. Srinivasan, "A Consumer-Based Methodology for the Identification of New Product Ideas," *Management Sci.*, 20 (1974), 921–937.

Syswerda, G., "Uniform Crossover in Genetic Algorithms," *Proc. Third International Conf. Genetic Algorithms*, J. D. Schaffer (Ed.), Morgan Kaufmann Publishers, San Mateo, CA, 1989, 2–9.

Zufryden, F., "A Conjoint-Measurement-Based Approach for Optimal New Product Design and Market Segmentation," in *Analytical Approaches to Product and Market Planning*, A. D. Shocker (Ed.), Marketing Science Institute, Cambridge, MA, 1977.

Accepted by Jehoshua Eliashberg, received September 30, 1993. *This paper has been with the authors* 10 *months for* 2 *revisions.*