Product Portfolio Evaluation Using Choice Modeling and Genetic Algorithms

Christopher N. Chapman, Microsoft
James L. Alford, Blink Interactive

DRAFT – for review purposes; final version to appear in Sawtooth Software Conference proceedings.

Abstract
We describe using genetic algorithm (GA) models to find near-optimal product portfolios in the presence of competition, using individual-level part worths from choice-based conjoint (CBC) and adaptive CBC (ACBC) models. We describe how to find optimal product portfolios, inform portfolio size, and generate hypotheses about product opportunities. Optimization routine is probabilistic and subject to data limitations and overfitting, so we bootstrap the process to find the expected distribution of likely outcomes across many resampled runs. We view this as an exploratory process to generate hypotheses in a product space; it is not a confirmatory or probative method. We offer the computer code necessary to run the GA portfolio model in R, given individual-level part worth estimates from another source (such as CBC/HB or ACBC).

Introduction

Like many marketing research organizations, at Microsoft Hardware we regularly collect information from discrete choice models to inform product design, engineering tradeoffs, and pricing. We routinely use choice-based conjoint analysis (CBC) and adaptive choice-based conjoint (ACBC) to inform our design decisions and category strategy. Given the success of those projects (e.g., Chapman et al, 2009; Chapman, Alford, and Love, 2009), we wondered whether conjoint analysis (CA) information could be used to inform higher-level strategic questions, such as whether we are making not only individually optimized products but also the optimal *number* of products within a category.

The general issue we examined was this: if we had insight into an entire portfolio – namely, the entire group of products that comprise a firm's offering in a category – what could we do with that information? These questions include: Are we making the right products? Are we offering too many products within a category (or not enough)? Are we differentiating our products effectively within the category? Are there potentially appealing products that we are not offering?

We sought a method that would help answer such questions. These issues are ultimately strategic in nature, so we did not seek to offer a conclusive empirical answer. Rather, we sought to develop a data-driven and reusable process that would inform strategy from an empirical point of view, and that could identify areas for further exploration and consideration for strategy and research. Additionally, we desired to design reusable and freely-available code so other analysts could explore the same questions with their data sets.

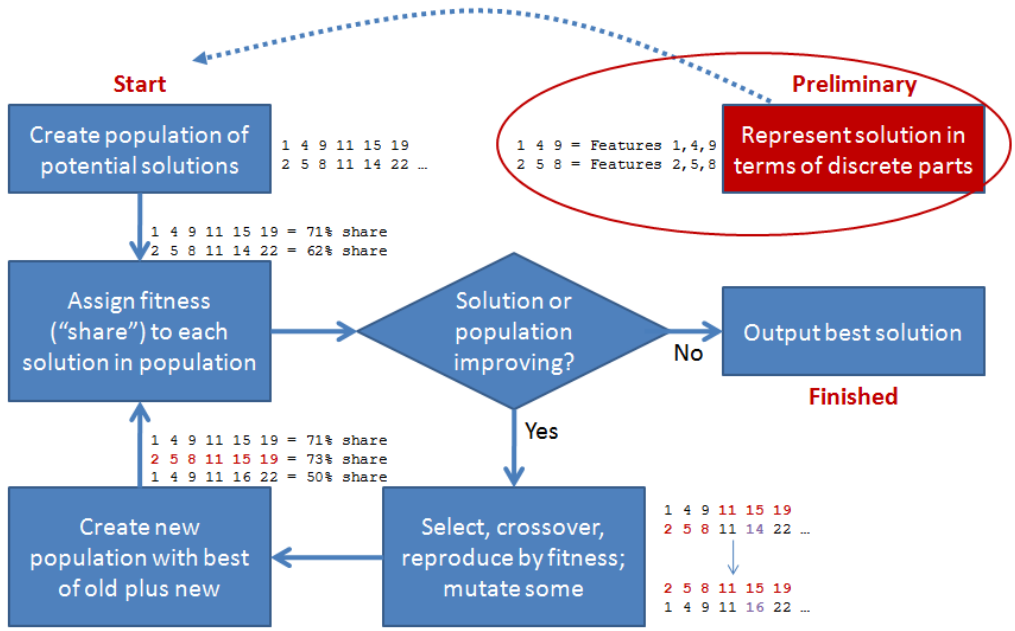
Method: Overview

To evaluate an existing product portfolio, we need a basis for comparison. A natural basis is to contrast the portfolio to an "ideal" portfolio. But how can one find an ideal portfolio? One possibility is to search the space of possible portfolios, and to evaluate each possible portfolio for customer appeal. To do this, one needs a method to search the portfolio space, which is likely to be large and complex, to evaluate each candidate portfolio, and to iterate towards an optimal solution.

Our process involved describing a candidate portfolio as a set of products, where each product comprises a set of attributes and features. Each candidate portfolio ("OURS") is evaluated against a set of other products ("COMP") that consists of current and anticipated competitive products. For each candidate portfolio we determined how many respondents would choose some product from OURS rather than one from COMP (or "None"). Each individual's preference within the portfolio was determined using individual-level HB part worths.

For a product space of any substantial complexity, there are too many portfolio possibilities to investigate through exhaustive search; a heuristic search method is needed to make the process computationally feasible. Belloni et al (2008) demonstrated that a genetic algorithm (GA) model is able to find near-optimal solutions to portfolio preference problems. We implemented such a GA method to select and optimize candidate portfolios. In that GA process, a population of potential portfolios is created initially at random; is evaluated and recombined to yield a new population; members of the population are selected according to fitness, mutated and/or recombined to yield a new population; and this iteration and recombination process is continued until additional improvement is unlikely. Figure 1 shows an outline of the GA process; we discuss the details of each stage below.

Figure 1: Schematic of GA Process (for a single run among many bootstrapped iterations)



With a single sample of data, GA models may overfit and capitalize on chance within the dataset. To counteract this, we implemented a bootstrapping approach, where a subset of respondent data is used to find a single "near-best" portfolio with one complete evolution of the GA search procedure (and, optionally, preference is itself bootstrapped within each evaluation cycle internal to the GA model). This process is conducted many times for different samples of respondent data, with each proposed near-best portfolio evaluated against holdout respondents not used in that iteration of the GA search. We then examine the distribution of results in the holdout evaluations, where we inspect the total portfolio result compared to other variables such as portfolio size, distribution of price points, features offered, and match to existing product offerings.

A single near-best portfolio may be obtained from the Sawtooth Software Advanced Simulation Module (ASM; Sawtooth Software, 2003). However, ASM does not run repeated samples with varying respondent sets and holdout respondents. Thus, to use ASM to examine the distribution of likely outcomes would require substantial manual work to resample respondents, re-run a model, and compile results. We believe the current version of ASM is especially valuable for single-product optimization and for basic exploration of portfolio models; depth exploration of portfolio distributions may be more easily conducted with a bootstrapped model as is presented here.

Detailed method

Genetic representation of portfolio

The first stage is to design a "genetic" representation of a possible solution, namely, a representation in which each functional element is represented as a discrete, replaceable part similar to a gene (Goldberg, 1989). In the present case, a single genetic solution is a portfolio of products, where each product is a list of features and attributes. Table 1 defines a product space with three attributes with 3-4 levels each.

Table 1: Product Attributes and Levels for a Simple CBC (example)

Attribute	***Levels***
1	1, 2, 3, 4
2	1, 2, 3, 4
3	1, 2, 3

For instance, "Attribute 1" might represent the feature "size," which occurs in four levels, while Attribute 2 could be "price," and so forth. A complete product is specified by choosing one level for each attribute.

A portfolio, then, is a collection of differentiated products specified in terms of their feature levels. Table 2 shows a possible lineup of products given the attributes and levels shown above.

Table 2: A Hypothetical Portfolio

Product	***Attribute 1***	***Attribute 2***	***Attribute 3***	***Etc.***
Product 1	Level 2	Level 1	Level 2	…
Product 2	Level 1	Level 1	Level 1	…
Etc. …				

After fielding a CA study, we typically have individual-level part worth estimates of the importance of attribute levels for each respondent. Those map directly to the list of attribute levels in successive columns within the part worth data, as shown in Table 3.

Table 3: Map of Features to Part Worth Data Columns

Attribute	***Levels***	***Part worth columns***
1	1, 2, 3, 4	1, 2, 3, 4

2	1, 2, 3, 4	5, 6, 7, 8
3	1, 2, 3	9, 10, 11

Each product is represented by the numbers of the columns that correspond to its feature levels. For instance, "Product 1" from Table 2 comprises features 2, 1, and 2 for its attributes, respectively, and those levels are represented by columns 2, 5, and 10 in the part worth data. Table 4 presents the column representation for the portfolio above.

Table 4: Portfolio Representation as Column Positions

Product	***Part worth columns***
Product 1	2, 5, 10
Product 2	1, 5, 9
…	

For the purpose of the GA model, this may be compacted into a single string in which successive products are simply compiled in order, as shown in Table 5.

Table 5: Genome Representation of Portfolio

Portfolio 1:	2, 5, 10, 1, 5, 9, …

With this representation, a portfolio is specified as a simple vector of integers. In this example, there are 3 integers per product such that a portfolio with 8 products would require 24 integers that represent the corresponding columns in the part worth data set.

Assessment of portfolio fitness

The GA method operates by finding a genetic representation – in this case, a vector of integers – that represents the best fit to some "fitness function." The key requirement for the analyst is to write this function for the problem at hand. Useful functions could be things such as the absolute preference that respondents have for a portfolio (i.e., likelihood to choose a product from it), or variants of preference such as maximized share vs. competition, revenue, or profit.

In the present study, the fitness function was designed to estimate the proportion of respondents who would choose any product from the portfolio, as opposed to choosing none. To estimate this, we evaluate the share of preference for each individual for each product, using the standard multinomial logit (MNL) model. The estimated "none" part worth is included. Each respondent is assigned by strict first choice to "prefer" the product (or "none") that receives the highest summed part worth score for its attributes. The fitness function then returns the proportion of respondents who choose any of the products in the portfolio as opposed to none. (This may be extended easily to consider competitive products by including them in a non-varying and non-evolving portion of the portfolio genotype.)

From the analyst's point of view, the most complex portion of the portfolio GA is writing and testing the appropriate fitness function that correctly evaluates a candidate portfolio and returns its value. Careful attention must be given to correct implementation of the MNL model and to appropriate pricing and feature prohibitions or interactions.

Genetic algorithm parameters
Given an appropriate fitness function, standard GA software may be used to find candidate solutions. We used the "rgenoud" package for the R statistics environment (Mebane and Sekhon, 2009; R Core Development Team, 2010), with parameters as detailed in Table 6.

Table 6: GA Parameters

GA library:	rgenoud (Mebane and Sekhon, 2009)
Genome structure:	1 integer per attribute (min=1, max= # of levels) * # attributes * portfolio size.
Population size:	400
Maximum generations:	50-200 (variable according to fitness trend)
Elitism:	Yes (best candidate always preserved)
Operators:	cloning; simple crossover; heuristic crossover; uniform mutation; boundary mutation; non-uniform mutation; and whole non-uniform mutation. (applied with equal odds across all reproduction events)

It is helpful to experiment with GA parameters in pilot runs. Initial exploration with our data set showed that improvement occurred rapidly within the first 50 generations of the GA model, so we specified 50 as the target number of generations unless recent improvement was shown. Likewise, experimentation with various population sizes showed that we needed at least 200 but no more than 600 population members, so we settled on a standard size of 400 members. Each member represents one complete, proposed portfolio.

Data sets, model iteration and bootstrapping
We used two datasets for this project: a CBC data set with N=716 respondents and an ACBC set with N=405 respondents. The attributes and features varied slightly between the CBC and ACBC data sets but concerned the same product category and had mostly identical feature levels. The CBC data had a total of 8 attributes with 34 feature levels, while the ACBC data had 9 attributes with 29 total levels. In our findings below, we compare the CBC and ACBC results for portfolio size, and then alternate consideration of the CBC and ACBC data for illustrative purposes.

The individual-level part worths came from CA studies implemented using Sawtooth Software SSI/Web (Sawtooth Software, 2010) with hierarchical Bayes (HB) estimation. The CA surveys were administered online to adult respondents in the US through a third-party panel provider. Individual-level part worths were estimated using Sawtooth Software CBC/HB and ACBC, respectively. For this present analysis, each individual's part worths were assigned to his or her within-respondent mean HB beta estimates. (Using individual HB draws instead of the respondent mean is an option in the available software code.)

For each data set (CBC and ACBC), we investigated different possible sizes of portfolio ranging from k=1 to k=20 products (specifically, k=1,2,4,6,8,10,12,16,20). At each size of portfolio, we run 50 iterations of the GA model to find 50 potential best portfolios. In those iterations, 60% of the respondents' individual-level part worths were sampled and used to evolve the GA, while 40% of the respondents were held out and used to evaluate the final result from the GA iteration.

For each proposed optimum portfolio, we recorded the portfolio size (number of products), the list of products and feature levels, and the preference share for each product and for none. The R statistics environment (R Development Core Team, 2010) was used as the analytic package after importing HB

estimates that were saved as a CSV file in Sawtooth Software SMRT. Generic GA functions were provided by the rgenoud package (Mebane and Sekhon, 2009), while the customized fitness function and utility code was written by the first author.

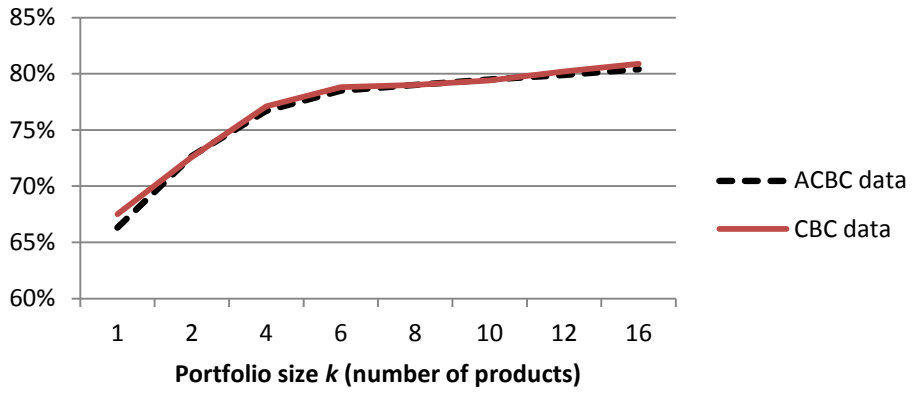
Findings

Portfolio size

Given the product attributes, how many products are needed to satisfy consumer demand? We addressed this by examining the proportion of people who would choose something other than "none," as portfolio size increased from 1 to 20 products (sampled 50 times for each portfolio size).

The median results using CBC and ACBC data appear in Figure 2, while the empirically observed credible intervals for CBC are shown in Figure 3.

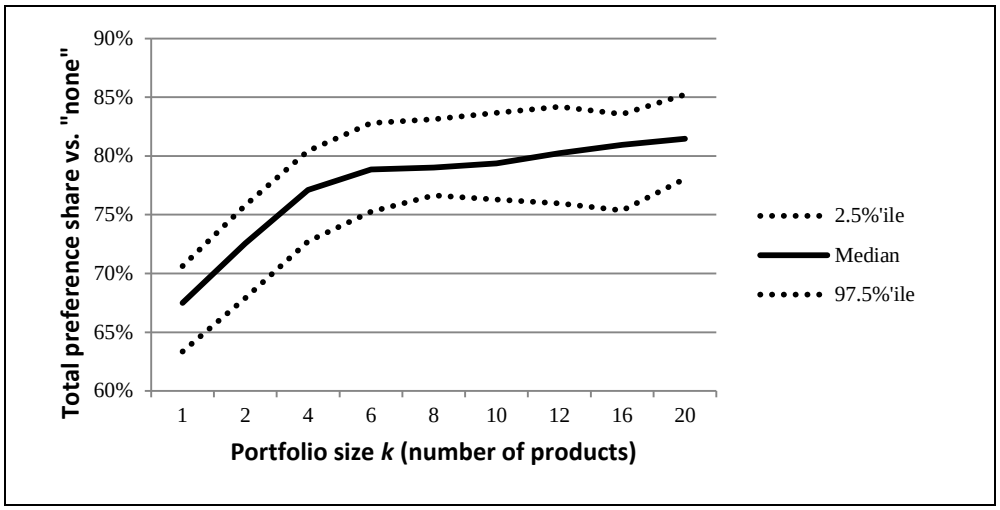
Figure 2: Median preference share met, by portfolio size



As shown in Figure 2, the incremental gain in preference share levels off sharply after k=6 products. Adding differentiated products continues to increase total preference, but each additional product above k=6 accounts for less than 1% increment in total-portfolio preference share.

It is striking that CBC and ACBC data show a virtually identical pattern of preference share by portfolio size; this gives a strong validation of the result with regards to internal model consistency.

Figure 3: 95% empirical credible intervals for preference share by portfolio size (CBC data)



In Figure 3, we see the observed credible intervals of preference share with L=50 runs at each portfolio size. The 95% intervals are approximately +/- 3% in preference share at each portfolio size. Above K=6 products, the lower bound of confidence never crosses the median estimate for K=6. This strengthens the observation that additional products are unlikely to satisfy substantial incremental demand (given the attributes and features studied).

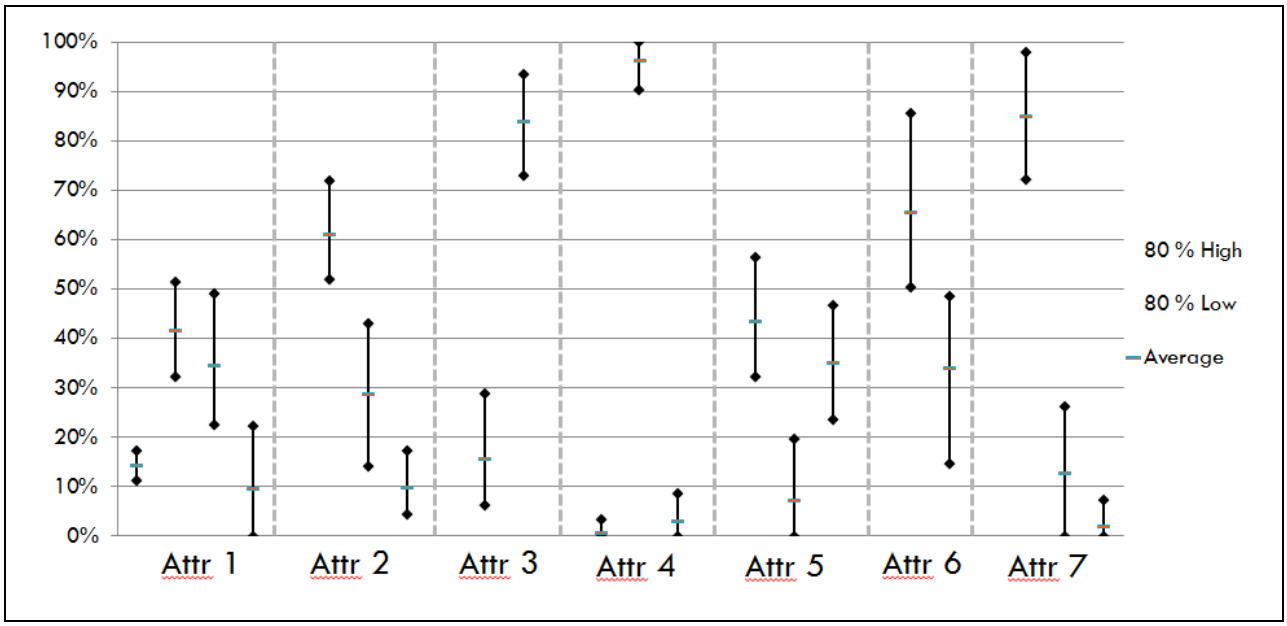
Within-portfolio preference share by feature

The portfolio data may be sliced by feature to examine feature demand. Although feature demand can be calculated quite easily from part worths using the multinomial logit (MNL) model, determining it on the basis of whole-portfolio demand has some advantages. First, feature co-occurrence can be examined easily (driven by respondent preference patterns). Second, the demand is bounded by portfolio size and thus may be forced to 0% or 100% more often than would be observed in part worths alone; this can be salient for managerial purposes. Third, it provides an alternate method of feature estimation in the presence of competition and boundaries, which may validate MNL estimation or provide another point of comparison to market data.

Feature level preference may be computed by examining each portfolio for each feature, and summing the preference of all the products within a portfolio in which that feature exists. For instance, suppose that in a k=6 product portfolio, 3 products have feature X. If the estimated demand of those 3 products is 5%, 4%, and 16%, respectively, then feature X would have an estimated demand of 5+4+16 = 25%.

Figure 4 shows 80% credible intervals for demand by feature level, across L=100 runs of portfolios with k=6 and k=8 products (ACBC data).

Figure 4: Preference share by feature in k=6-8 product portfolios



In Figure 4, we see that many features have quite wide ranges of estimated demand, indicating that they are of interest to consumers but may be managed with relatively great latitude within a portfolio. Other features, however have almost zero demand (e.g., Attribute 4-Feature 1) or universal demand (Attribute 4-Feature 2).

In addition to investigating demand, we have compared these estimates to actual market data. Those findings cannot be reported in detail due to confidentiality and market data restrictions, but one comparison may exemplify them. Attribute 2-Feature 2 is a relatively unique feature whose market demand is of substantial interest to our product team. Its current market penetration is approximately 35% (+/- 4%) according to a recent, separately fielded survey by our team. We see in Figure 4 that Attribute 2-Feature 2 has an estimated portfolio-basis demand of approximately 14%-43%, with a median estimate of 29%. The median estimate is quite close to the actual market penetration and the range overlaps the actual value; thus the portfolio estimate is consistent with performance that we should expect in the market.

This kind of estimate may be useful as a diagnostic indicator. For instance, if a feature were performing worse in the market than was expected in the portfolio model, one could consider targeting it for increased communication or other market intervention.

Product opportunities

To find potential opportunities, one may examine the products that often appear across the optimal portfolios. A simple way to do this is to count the number of times that a fully-specified product (i.e., a complete product string) appears across portfolios.

Table 7 lists the products that appeared in more than 20% of CBC portfolio runs (excluding runs with K=1 or 2 products). Seven products appeared very commonly, of which two are currently not available in the market (lines 3 and 6 in Table 7). Comparing those products to the others, the unique feature of those two products is the combination of attribute 2/level 2 (“x2xxxxx” in the feature code list) occurring with attribute 6/level 2 (“xxxxx2x” in the list).

Table 7: Proportion of times a given product appears in a portfolio with K≥4 products
(currently unproduced products shown in bold)

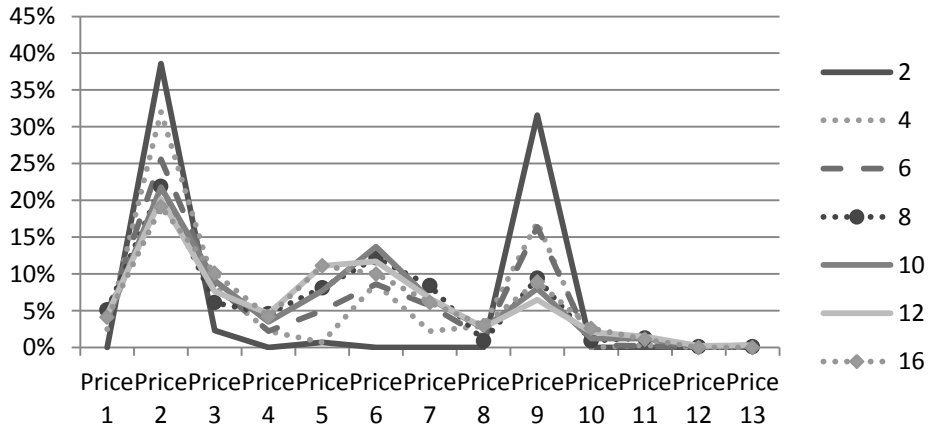
	Proportion of all portfolios (N=800, K≥4)	Feature codes (excluding price)
1	0.76	2111112
2	0.47	1311512
3	**0.45**	**3211422**
4	0.26	1121512
5	0.23	2111111
6	**0.22**	**3211122**
7	0.21	3111412

We used this information to investigate the feasibility and cost of combining those attributes in a new product. More generally, we find that this type of investigation can be an easy way to generate product ideas with existing data. Those ideas must be subjected to further vetting and confirmation, but we believe it is useful to have an automated procedure of this kind to generate ideas along with initial supporting data.

Price bands
The fitness algorithm requires pricing information (if relevant) to determine the portfolio composition. By inspecting the price distribution of products found in optimal portfolios, one may form hypotheses about consumer price expectations.

Figure 5 shows the occurrence of products in optimal portfolios, counting the occurrence of products at given price points. Price 1 to Price 13 span the common (but not entire) range of product pricing in the category of interest. When only two products are produced, the most common price points are Price 2 and Price 9. As more products are produced, the distribution of price points becomes more diffuse, yet there are still obvious peaks around Price 2, Price 6, and Price 9.

Figure 5: Product distribution by price and portfolio size (K=2-16, ACBC data)



These results raised several ideas for the product team. First, they call into question a highly stacked pricing approach that attempts to offer product at many different price points. The results suggest that it may be better to offer products at fewer price points but with more differentiation (e.g., along the lines of the opportunities identified in the previous section).

The findings suggest that there is little demand for very high-priced products in this category (prices above Price 9) when the portfolios are otherwise near optimal. Thus, the existence of high-priced products in the market may be due to inefficiencies in portfolio offerings rather a consumer desire for fully-loaded products as such.

The results also suggest that most consumers are willing to pay slightly more than Price 1 if they are offered a feature of interest. This information was of great interest to retail partners. Again, the information is exploratory, not definitive, but is nevertheless useful because it is easily available from this process applied to existing data, while it might be difficult or impossible to obtain otherwise.

Discussion

There are several limitations of this method and areas for exploration. The primary limitation is that this process is only as good as the data provided. It assumes that one has collected information correctly, with an appropriate sample and CA design, for attributes that accurately and completely define a product space. None of those is a simple requirement. If important attributes are omitted, then the results will be difficult to interpret at best, or misleading at worst. Thus, it is important to perform exploration of this kind in a space that one understands well, or at the very least, to interpret the results cautiously and as provisional findings.

As we have noted repeatedly in this paper, the process is primarily exploratory. We believe it provides useful insight and generates hypotheses from existing data with relatively little effort and in a way that is complementary to other approaches. When implemented carefully, such results may be better than having no information, but all implications should be confirmed or checked with other methods.

There are many open questions about methods of this kind and the relationship between the stochastic operation of GAs (and other search algorithms) and the assumptions of the underlying data that come from HB and other preference estimation processes. For instance, it is conceivable that assumptions of HB (e.g., regarding distributions, error structure, and interactions) could interact with the search method in such a way that portions of the results are structured by the estimation process itself rather than by respondents' data. It may be very difficult either to confirm or dispute that possibility. We believe this is an important area for academic exploration; and in the intervening time, is yet another good reason to regard the output of this method as exploratory and generative in nature.

Another area for future research concerns the relationship among search methods and alternative ways to represent a portfolio. For instance, one might consider using search algorithms other than a GA (see Belloni et al, 2008) or portfolio methods derived from quantitative finance.

Computer Code

Computer code is available from the first author. It is free, open source, use-at-your-own-risk, and unwarrantied code provided solely for didactic and research purposes. The code implements the bootstrapped GA model for applications that provide standard HB utilities, such as the typical output of

HB estimation in a Sawtooth Software CBC, ACBC, or similar conjoint analysis project. It is written in R and requires modest customization to the fitness function to implement pricing and attribute prohibitions or interactions (if any). We estimate that adaptation of the code to a given project should take approximately one day of analyst time, if the analyst has basic familiarity with the R language.

Options in the code include the ability to use either first choice or aggregate share of preference estimation; options to use HB draws instead of individual mean part worths; bootstrapping within the fitness function itself (e.g., across HB draws); and logit model exponent tuning. The default fitness function implements a share of preference model for portfolio fitness, but this could be adapted to implement fitness in terms of revenue, profit, directed competition, or other metrics.

An analyst may wish to explore optimization procedures other than a GA. In that case, it would be possible to use the provided framework to handle CA data and fitness assignment, but to replace the call to rgenoud that performs GA optimization and use another optimization routine instead. The primary requirement in that case would be to write an appropriate wrapper function that calls the optimization routine.

GA models are computationally intensive, and a large-scale project such as the one reported here may take days or weeks to run. The key aspects that increase run time are portfolio size, number of generations in the GA, and the GA population size; each of those produces a nearly linear increase in time. Computation time may be reduced either by simplistic brute force parallelization, such as running the model for different portfolio sizes on different computers; or by using a multicore workstation with parallelization options for the rgenoud library. The latter approach requires additional configuration of the R environment (cf. Mebane and Sekhon, 2009).

Conclusion

Search and optimization methods such as the GA approach offer analysts a way to mine existing data and derive insight along with potential opportunities for products. The method presented here offers a straightforward method to do this with HB data from discrete choice studies. The authors hope that the approach and the available code are useful to others, and we look forward to seeing future work from the choice modeling community.

Acknowledgements
The authors thank Ken Deal of McMaster University for insightful comments on the proposal, computer code, and presentation. Bryan Orme of Sawtooth Software commented on the approach and its relationship to other methods. Attendees of the 2010 Sawtooth Software Conference provided many interesting questions, comments, and related approaches; in particular, the audience observed how the fitness function could be scoped to different needs, and how the search algorithm itself could use a process other than a genetic algorithm. We have attempted to reflect those comments in this paper; oversights and errors are, of course, our own.

Author contact
Questions about the computer code and other issues should be directed to Chris Chapman: chris.chapman@microsoft.com (primary), or cnchapman@msn.com (backup). Mailing address: Chris Chapman (cchap), 1 Microsoft Way, Redmond, WA 98052.

References

Belloni, A., Freund, R.M, Selove, M., and Simester, D. (2008). Optimal Product Line Design: Efficient Methods and Comparisons. *Management Science* 54: 9, September 2008, pp. 1544-1552.

Chapman, C.N., Alford, J.L., and Love, E. (2009). Exploring the reliability and validity of conjoint analysis studies. Presented at Advanced Research Techniques Forum (A/R/T Forum), Whistler, BC, June 2009.

Chapman, C.N., Alford, J.L., Johnson, C., Lahav, M., and Weidemann, R. (2009). Comparing results of CBC and ACBC with real product selection. *Proceedings of the 2009 Sawtooth Software Conference*, Del Ray Beach, FL, March 2009.

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

Mebane, W.R., Jr., Sekhon, J.S. (2009). Genetic Optimization Using Derivatives: The rgenoud Package for R. [R package], http://sekhon. berkeley.edu/rgenoud.

R Development Core Team (2010). R: A language and environment for statistical computing [Computer software]. Version 2.11.0. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org.

Sawtooth Software (2003). *Advanced Simulation Module for Product Optimization v1.5 Technical Paper*. Sequim, WA.

Sawtooth Software (2010a). CBC/HB 5.0 [Computer software.] Sequim, WA, 2010.

Sawtooth Software (2010b). SSI/Web 7.0 [Computer software.] Sequim, WA, 2010.