“The Various Brands in the Optimal Product Line vs. the Optimal Single Product”

AUTHORS	Gila E. Fruchter Ariel Fligler

ARTICLE INFO	Gila E. Fruchter and Ariel Fligler (2007). The Various Brands in the Optimal Product Line vs. the Optimal Single Product. *Innovative Marketing* , *3*(1)
RELEASED ON	Thursday, 08 February 2007
JOURNAL	"Innovative Marketing "
FOUNDER	LLC “Consulting Publishing Company “Business Perspectives”

NUMBER OF REFERENCES	NUMBER OF FIGURES	NUMBER OF TABLES
0	0	0

THE VARIOUS BRANDS IN THE OPTIMAL PRODUCT LINE VS. THE OPTIMAL SINGLE PRODUCT

Gila E. Fruchter[*], Ariel Fligler[**]

Abstract

In this marketing-oriented era where manufacturers maximize profits through customer satisfaction, there is an increasing need to design a product line rather than a single product. By offering a product line, the manufacturer can customize his or her products to the needs of a variety of segments in order to maximize profits by satisfying more customers than a single product would. Recently, the authors introduced a genetic algorithm to solve this problem. An interesting product design research question is how the profile of an optimal product line differs from an optimal single product? To analyse this question we apply our method to a single product design problem. Surprizingly, we find that the profile of the optimal single product is the core product of the optimal product line. Furthermore, the various brands in the product line are slight variations of the single product solution. An important managerial implication of this property could be designing a core brand with a number of line extensions.

Key words: genetic algorithms, heuristics, product line design, cannibalization, marketing.

1. Introduction

In this marketing-oriented era where manufacturers maximize profits through customer satisfaction, there is an increasing need to design a product line rather than a single product. By offering a product line, the manufacturer can customize his or her products to the needs of a variety of segments in order to maximize profits by satisfying more customers than a single product would. We define a product to be a set of attributes where each attribute can have different levels. Thus, a product line will be a set of different attribute level configurations.

In offering a product line to achieve maximal profit, the manufacturer needs to carefully choose the variety of product configurations and their prices given the products' costs and customers' willingness to pay. The main problem in designing a profit maximizing product line is to target the 'right product' to the 'right customer'. A customer wants to optimize the utility from purchase by choosing a product that maximizes his surplus. In contrast, the manufacturer wants to maximize profits by extracting the customer's surplus. However, in an open market, this strategy leads to cannibalization where customers with high willingness to pay may choose products designed for customers with low willingness to pay. The classical approach to avoid cannibalization is to reprice the product configurations incorporating the customer's willingness and wants. This method is possible when we have a small set of product configurations and customer segments or alternatively when analytical relations can be established between customer preferences and product configurations. However, in reality, when the amount of data on customer preferences or possible product configurations is large and no analytical relations can be established, the problem of an optimal product line design becomes very difficult and there are no traditional methods to solve it. In fact, the product line design problem has been shown to belong to a group of problems known in computer science as NP-Complete (Kohli and Krishnamurti, 1989). Though it has not yet been proven, it is assumed that NP-Complete problems have a time complexity that is an exponential function of the problem size. Thus, usage of heuristics or a nondeterministic algorithm is a common practice for their solution.

The problem of product design has been heavily studied in the literature over the past 20 years. Heuristics to solve the single product design problem have been proposed by Kohli and Krishna-

[*] Graduate School of Business Administration, Bar-Ilan University, Ramat-Gan, Israel.
[**] Olista, Netanya, Israel.

murti (1987) using a dynamic programming approach and by Balakrishnan and Jacob (1996) using a genetic algorithm. Balakrishnan and Jacob solved the product design problem using a genetic algorithm for the share of choices, seller's return and buyer's welfare objectives working with part-worth data. They report their results on a set of randomly generated datasets and compare the genetic algorithm's performance with the dynamic programming heuristic of Kohli and Krishnamurti. Their work showed the advantage of the genetic algorithms over the dynamic programming approach in solving the problem of single product design.

Green and Krieger (1985) were among the first to model the product line design problem. They presented a heuristic algorithm to solve the seller's return and buyer's welfare problems. McBride and Zufryden (1988) described a method to solve the product line design problem using an Integer Programming method with a model that follows Green and Krieger's. Dobson and Kalish (1993) presented a heuristic-based approach to the product line design problem that extended the model developed by Green and Krieger. Kohli and Sukumar (1990) followed a dynamic programming-like approach using a model similar to Green and Krieger's. Kohli and Sukumar's ideas were extended by Nair et al. (1995) who used a beam search method that is based on a breadth-first search algorithm without backtracking. Alexouda and Paparizos (2001) have used Kohli and Sukumar (1990) mathematical formulation to test a method using a genetic algorithm. They showed that genetic algorithms have significant potential to solve product line design problems.

Thus, there are two different approaches to attack the product line design problem. The first approach considers that first a finite set of reference products is obtained from which a product line is selected (Green and Krieger, 1985; McBride and Zufryden, 1988; and Dobson and Kalish, 1993). If the number of attributes and attribute levels is large and the most attribute level combinations define feasible products, it can be computationally infeasible to enumerate the utilities of candidate items. For this class of problems, it is preferable to use the second approach, which constructs product lines directly from parth-worth data. The methods proposed by Kohli and Sukumar (1990), Nair, Thakur and Wen (1995) and Alexouda and Paparrizos (2001) use the second approach.

Recently, Fruchter et al. (2006), employed the idea of genetic algorithms to the problem of optimal product line design, considering the first approach. Refraining from working directly with part-worth data in the chromosome genes, they succeeded in having shorter chromosomes and no dependency on products' utility evaluation method, which leads to a more effective search process. Special domain operators were developed to help the genetic algorithm mitigate cannibalization and enhance the algorithm's local search abilities. Using manufacturer's profits as the criteria for fitness in evaluating chromosomes, the usage of domain specific operators was found to be highly beneficial with better final results. They also have hybridized the genetic algorithm with a linear programming post-processing step to fine tune the prices of products in the product line. Attacking the core difficulty of canibalization in the algorithm, the operators introduced in this work are unique.

In this article, we are interested to find how the profile of the various brands of a product line differs from an optimal single product. To see this we apply the algorithm to a single product design problem. Surprisingly, we find that the profile of the optimal single product is the core product of the optimal product line. Furthermore, the various brands in the product line are slight variations of the single product solution. To check the effectiveness of the methodology, in the sequel, we present its performance evaluation.

The rest of the paper is organized as follows. In Section 2 we apply the algorithm to a single product design problem. In Section 3 we present the performance evaluation, and in Section 4 we summarize our findings.

2. Application of the Methodology of Fruchter et al. (2006) to the Single Product Design Problem

Applying the methodology of Fruchter et al. (2006), for the single product design, for the real and simulated data sets (see Fruchter et al., 2006), for problem sizes of 30, 150 and 250 products, produces the results described in Table 1.

Table 1

The Single Product vs. the Product Line Design Problem

		Simulated 30	Simulated 150	Simulated 250	Real
Profits	Product Line	93,769	11,143,001	50,820,037	12,431
	Single	84,000	10,100,000	46,574,630	5,434
Market Served	Product Line	29	149	248	32
	Single	21	101	167	19
Average Customer's Surplus	Product Line	1,640	47,206	140,270	82
	Single	2,190	50,990	153,898	108

As can be seen, both profits and the market served are greater, in all runs, by offering a product line instead of a single product. The average customer surplus is lower in the product line offer than in the single product offer. The smaller average surplus can be attributed to the efficiency of the product line approach in extracting surpluses. In the product line approach, the manufacturer can offer the market several products, each tailored to fit the needs of the individual customer, and thus more customers are buying products. As products better fit the needs of customers, customer's willingness to pay is higher and consequently the manufacturer can set a higher price and gain higher profits. When offering a single product, the manufacturer must balance the needs of various customers with the single optimal product with respect to profits, and thus might not target all customers. One can see that manufacturer's profits gained by the product line approach are higher in the real data compared to the simulated one. This can be attributed to internal differences in structures of the two data sets. The simulated data set (see Fruchter et al., 2006) is built so that every customer has a positive reservation price for every product and thus every product has some complementary product for that customer. Thus, when moving from a product line to a single product, more customers in the simulated data can find the single product appealing and buy it. Thus the decrease in the number of buying customers in the simulated data set is to a lesser extent than in the real data set. Furthermore, the simulated data set has a more symmetrical internal structure as reservation price graphs of customers follow the same direction and differ only in their inclination. This internal structure can explain why the number of customers and manufacturer's profits degrade in almost the same extent in different problem sizes of the simulated data.

The optimal single product profile and the appearance of dominant attributes levels in the optimal product line are described in Table 2. As we can see 4 out of the 6 dominant attribute levels appear in the optimal single product profile. Thus, the conclusions of our research can be summarized as follows:

(i) The profile of the optimal single product is the core product of the optimal product line.
(ii) The various brands in the product line are slight variations of the single product solution.

This could imply, for example, a core brand with a number of line extensions.

Table 2

The Single Product Profile and the Appearance of Attribute Levels in the Product Line

	Memory			Processor			HD		Modem		CD		Screen	
	32M	64M	128M	400 MHz	500 MHz	650 MHz	6G	12G	Y	N	Y	N	High	Low
The Single Product Profile (50[1])			✓	✓			✓			✓	✓		✓	
Appearance In the Optimal Product Line	**13%**	**60%**	**27%**	**53%**	**33%**	**14%**	**66%**	**24%**	**14%**	**86%**	**46%**	**54%**	**87%**	**13%**

Note: [1] Product 50 (compared with Fruchter et al., 2006) is the solution for the Single Product Design problem.

To check the effectiveness of the methodology, in the next section, we present its performance evaluation.

3. Performance Evaluation

In this section, we evaluate the performance of the proposed method in Fruchter et al. (2006) by means of: (i) sensitivity analysis of domain specific operators' usage, (ii) assessing the linear programming optimization step advantage, and (iii) comparison with the known heuristic of Dobson and Kalish (1993).

Sensitivity Analysis of Domain Specific Operators Usage

We first analyze the influence of problem domain operators which have been specially developed to solve the product line design problem by a genetic algorithm and are not part of the algorithm's repertoire in general. We chose parameters like profit, market served and product line length to analyze the performance. To assess this influence we tested the algorithm with operators being activated individually, or with all three operators used in tandem. Our results are described in Table 3. As Table 3 clearly shows, usage of any operator by itself generated better results compared with running the algorithm without the operators. Furthermore, the combined usage of all operators generated better results compared with running the algorithm with each operator used individually. Following Table 3, in the case of real data set (for details see Fruchter et al., 2006), the usage of Cannibalization Avoidance and Market Diversification operators created the highest impact. As cannibalization was noted as a major difficulty in pricing a product line in a heterogeneous market, the results support the usage of such an operator to mitigate this problem. The Market Diversification operator's importance results from the genetic algorithm's stochastic nature. Due to mutation's random changes and the possible destructive behavior of the crossover procedure, products are constantly removed from the product line. Furthermore, genetic drift causes gene locations to become homogenous and combinations of genes representing a certain product might not exist in the chromosome population. As the number of customers and possible product configurations grow, the Rejuvenation procedure might not compensate for this problem. Thus, a deterministic introduction of products into the product line enables the genetic algorithm to create better gene combinations without relying on the standard algorithm's stochastic dynamics.

Table 3

The Effect of Domain Specific Operators on GA's Performance

Combination	Simulated Data					
	Profit	No. Of Products (Out of 150)	Satisfied Customers (Out of 150)	Aggregated Final Customers' Surplus	Generations	Time (Seconds)
No Operators	1.1081E7	17	149	7746035	685	2400
Market Diversification Operator	1.1074E7	18	150	7833019	790	3600
Cannibalization Avoidance Operator	1.1101E7	20	149	7457785	936	8759
Post-Processing Linear Programming Optimization Operator	1.0701E7	17	150	8593510	366	4235
Price Improvement Operator	1.1106E7	15	149	7334095	725	2520
All Operators	1.1210E7	14	148	7275002	598	7547

Table 3 (continuous)

Combination	Real Data					
	Profit	No. Of Products (Out of 144)	Satisfied Customers (Out of 61)	Aggregated Final Customers' Surplus	Generations	Time (Seconds)
No Operators	10031	6	26	3224	216	138
Market Diversification Operator	11387	12	29	2865	342	564
Cannibalization Avoidance Operator	11482	12	30	3252	211	4800
Post-Processing Linear Programming Optimization Operator	10261	7	21	3073	71	8640
Price Improvement Operator	10484	8	28	3432	133	1200
All Operators	12178	10	31	4438	200	5400

The heuristic and Linear Programming based Price Improvement operators' impact on results is of a lesser extent than the Cannibalization Avoidance and Market Diversification operators. This can be attributed to their local behavior, as they do not change market response and their influence is restricted to price changes for a given assignment of products to customers. An interesting point is the fact that the heuristic Price Improvement operator generated better results than the LP version. A clue for this behavior can be found in the lower generation count of 71 generations when the LP version was used. Since the LP version finds optimal pricing for a given assignment, in most cases, the operator with the same price settings will transform two chromosomes with the same assignment but with a different price setting. Thus, the two chromosomes will become identical to each other. Even though the LP version yielded improvements in profits of up to 90% on a single chromosome, the genetic algorithm converges quickly into a homogeneous population before it has enough time to find better solutions. Thus, we have used the Price Improvement operator only to optimize the final solution generated by the genetic algorithm. For the same reason, we have refrained from using the LP Price Improvement operator to optimize the initial chromosome population.

Table 3 shows that the effect of the domain operators' usage was smaller on the simulated data than on the real one. We can attribute this difference to the internal structure differences between the two data sets. Specifically, in the simulated data set every customer has a positive utility from every product configuration. Thus the impact of a product drawn out of the product line on the number of buying customers is lower. The net effect is that the Market Diversification operator's impact on performance is negligible on the simulated data set.

Contrary to that, the impact of the Price Improvement operators on the simulated data set was higher than on the real one. We can attribute this finding to the inclination of customers' reservation price graphs in the simulated data set. As further one goes along the products (X) axis, the differences between consecutive customer's reservation prices from a product become bigger. Thus, the Price Improvement operators have more room to enlarge product prices, yielding an increase in profits.

The combined usage of all domain operators generated the best results, showing the synergy between them. Although the operators' usage has implications on performance, the increase of 21% in the final profits justifies in our mind their usage.

Assessing the Post Processing Linear Programming Stage

We have used linear programming optimization to develop our LP Price Improvement operator. As described in the previous section, using the LP Price Improvement operator in every generation of the algorithm caused the algorithm to converge prematurely and had severe performance implications. Thus, we have elected to use our heuristic operator in the deterministic step instead. Since the heuristic operator does not guarantee an optimal price setting, we have used the LP version in the post-processing step to fine-tune the best result gained by the algorithm.

The improvements gained by running the linear programming optimization procedure on the genetic algorithm's final solution are described in Table 4. With domain operators used and inspecting a converged GA population, a difference of 2% was gained in favor of the linear programming method. As can be seen, the extent by which the linear programming optimization enhanced the genetic algorithm's final solution depends on the maturity of the solution and the usage of domain specific operators. Larger enhancements were gained when run on genetic algorithm solutions generated after fewer generations. Product pricing in chromosomes generated after many generations is more mature since the algorithm has more time to build good pricing combinations in the price genes.

Table 4

LP Improvements to a Genetic Algorithm's Final Solution for Solutions Generated after Different Number of Generations

Genetic Algorithm Run Without Domain Operators			
GA Generation	*GA Result*	*LP Result*	*Improvement (%)*
5	6339	9390	45%
50	9086	10401	14%
200	9287	10503	13%
500	10334	11192	8%
Genetic Algorithm Run With[2] Domain Operators			
GA Generation	*GA Result*	*LP Result*	*Improvement (%)*
5	7484	8604	15%
50	9842	10833	10%
200	10771	11243	4%
500	11198	11444	2.2%

Note: [2] The heuristic *Price Improvement* operator has been used.

Furthermore, when domain specific operators have been used, the enhancements generated with the linear programming method were even smaller. We attribute this to the enhancements gained by the heuristic Price Improvement operator. When a genetic algorithm's solution was generated by a run when no domain operators were used, it was optimized separately by the heuristic Price Improvement operator and the linear optimization post processing step.

To summarize, we see the linear programming option as a beneficial addition to the genetic algorithm in fine-tuning the genetic algorithm's final solutions, compensating for its stochastic nature.

Comparative Results with Dobson-Kalish Heuristics

Dobson and Kalish (1993) describe a greedy heuristic to solve the product line design problem using a problem model similar to ours. To assess the performance of our methodology, we next

compare the GA developed in Fruchter el al. (2006) for both the simulated and real data sets (see Fruchter et al., 2006) vis-à-vis the Dobson-Kalish heuristic.

The heuristic is based on an iterative greedy approach where in each iteration, the heuristic seeks to include one more product in the product line. The product, which is added to the product line, is the one, which maximizes the increase of the manufacturer's profits. The heuristic as described in Dobson and Kalish (1993) works as follows.

Step 1: An initial valid assignment of products to customers is generated at random and the optimal prices for the assignment are calculated.

Step 2: All products are ranked by total utility contribution $\sum_{i\in\{1,\ldots I\}} u_{ij}$.

Step 3a: Brand *j* is added to the product line with price "infinity".

Step 3b: The price of brand *j* for customer *i* is calculated by:

$v_{ij} = p_{k(i)} + (u_{ij} - u_{ik(i)})$, where $k(i)$ is the currently bought brand by customer *i*.

Step 3c: All v_{ij} are sorted in descending order.

Step 3d: The profit increase by adding product *j* with price v_{ij} is calculated for every $v_{ij} > 0$. The product that yields the highest profit increase is a new candidate.

Step 3 is repeated for every product till no further improvement is obtained. When adding a product reduces profits, the heuristic stops. We ran the heuristic 30,000 times until the best performance was obtained on our data set.

In Table 5 we compare the results obtained by Dobson-Kalish heuristic and our GA by considering three quantities: profits, the market served and the aggregated customer surplus. As we can see, the genetic algorithm outperforms the heuristic on both the simulated and real data sets. Furthermore, when we used the best result generated by our genetic algorithm as an initial solution for the heuristic's initialization Step 1, no further improvements in profits were gained.

Table 5

Dobson-Kalish Heuristic vs. Genetic Algorithm

		Simulated 250	Real
Profits	Genetic Algorithm	50,820,037	12,364
	Dobson-Kalish Heuristic	49,756,040	8,764
Market Served	Genetic Algorithm	248	32
	Dobson-Kalish Heuristic	250	29
Aggregated Surplus	Genetic Algorithm	38,166,797	2,653
	Dobson-Kalish Heuristic	41,959,233	3,005

From Table 5 we can see that the heuristic performs better on the simulated data set, generating a result lower in 3% than the genetic algorithm's solution, versus a result lower in 29% on the real data set. We can attribute this behavior to the internal structure of the simulated data set where products have a strict welfare order between themselves and products with higher welfare values are the ones who constitute the optimal product line. This parallels the welfare ranking order by which products are added to the product line in the heuristic.

We also tested the appearance rates of various product attributes levels for the genetic algorithm and heuristic's best results on our data set. We describe this in Table 6. It can be seen that in 66% of the attributes, the ratio of appearance rates between attribute levels in the heuristic and in the genetic algorithm's solutions were the same. This fact strengthen our observation of a core attributes levels group which are common across products in the optimal product line.

Table 6

Comparative Results of the Attribute Levels Appearance

	Memory			Processor			HD		Modem		CD		Screen	
	32M	64M	128M	400 MHz	500 MHz	650 MHz	6 GB	12 GB	Y	N	Y	N	High	Low
Appearance Heuristic	30%	40%	30%	40%	40%	20%	60%	40%	90%	10%	90%	10%	70%	30%
Appearance Genetic Algorithm	13%	60%	27%	53%	33%	14%	66%	24%	14%	86%	46%	54%	87%	13%

4. Conclusions

The recently introduced methodology in Fruchter et al. (2006) presents a genetic algorithm specifically tailored to solve the product line design problem. The algorithm was implemented on a real world scenario. The algorithm was tried on both simulated and real data sets. Comparing the Fruchter et al.'s (2006) algorithm with a known state of the art heuristic (Dobson and Kalish, 1993) showed its superiority. Applying the algorithm to a particular product line problem design, the single product design, we find a very interesting feature of the product line design problem. We find that there exists a core of high quality attribute levels. These high quality attributes' levels, appear with high probability in brands that are part of the final solution of both single product and product line. An important managerial implication of this property could be designing a core brand with a number of line extensions.

References

1. Alexouda G. and K. Paparizos, 2001. "A Genetic Algorithm Approach to the Product Line Design Problem Using the Seller's Return Criterion: An Extensive Comparative Computational Study", *European Journal of Operational Research*, 134, 165-178.
2. Balakrishnan P.V. and V.S. Jacob, 1996. "Genetic Algorithm for Product Design", *Management Science*, 42, 1105-1117.
3. Blickle T. and L.Thiele, 1995. "A Comparison of Selection Schemes Used in Genetic Algorithms", *TIK-Report No. 11, Swiss Federal Institute Of Technology* .
4. Dobson G. and S. Kalish, 1993. "Heuristics for Pricing and Positioning a Product Line Using Conjoint and Cost Data", *Management Science*, 39, 160-175.
5. Dobson G. and S. Kalish, 1988. "Positioning and Pricing a Product Line", *Marketing Science*, 7, 107-125.
6. Fligler, A., 2003. "Product Line Design Using a Genetic Algorithm", *Thesis*, Technion Library.
7. Fruchter, G.E., A. Fligler and R.S. Winer, 2006. "Optimal Product Line Design: Genetic Algorithm Approach to Mitigate Cannibalization", *Journal of Optimization Theory and Application*, 131, 227-244.
8. Goldberg E.D., 1989. "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley.
9. Green P.E. and A.M. Krieger, 1985. "Models and Heuristics for Product Line selection", *Marketing Science*, 4, 1-19.
10. Holland H.J., 1992. "Adaptation In Natural And Artificial Systems", MIT Press.
11. Kohli R. and R. Krishnamurti, 1987. "A Heuristic Approach to Product Design", *Management Science*, 33, 1523-1533.
12. Kohli R. and R. Krishnamurti, 1989. "Optimal Product Design Conjoint Analysis: Computational Complexity and Algorithms", *European Journal of Operational Research*, 40, 186-195.
13. Kohli R. and R. Sukumar, 1990. "Heuristics For Product Line Design Using Conjoint Analysis", *Management Science*, 12, 1464-1478.
14. McBride R.D. and S.F. Zufryden, 1988. "An Integer Programming Approach To the Optimal Product Line Selection Problem", *Marketing Science*, 7, 126-140.
15. Mitchell M., 1996. "An Introduction to Genetic Algorithms", MIT Press.
16. Nair, S.K., L.S. Thakur, and K. Wen,, 1995. "Near Optimal Solutions For Product Line Design And Selection: Beam Search Heuristics", *Management Science*, 41(5), 767-785.