Investigating Machine Learning Techniques for Solving Product-line Optimization Problems

Sascha Voekler and Daniel Baier

Abstract Product-line optimization using consumers' preferences measured by conjoint analysis is an important issue to marketing researchers. Since it is a combinatorial NP-hard optimization problem, several meta-heuristics have been proposed to ensure at least near-optimal solutions. This work presents already used meta-heuristics in the context of product-line optimization like genetic algorithms, simulated annealing, particle-swarm optimization, and ant-colony optimization. Furthermore, other promising approaches like harmony search, multiverse optimizer and memetic algorithms are introduced to the topic. All of these algorithms are applied to a function for maximizing profits with a probabilistic choice rule. The performances of the meta-heuristics are measured in terms of best and average solution quality. To determine the most suitable meta-heuristics for the underlying objective function, a Monte Carlo simulation for several different problem instances with simulated data is performed. Simulation results suggest the use of genetic algorithms, simulated annealing and memetic algorithms for product-line optimization.

Sascha Voekler ✉ sascha.voekler@b-tu.de
Brandenburg University of Technology

Daniel Baier ✉ daniel.baier@uni-bayreuth.de
University of Bayreuth

Archives of Data Science, Series A
(Online First)
KIT Scientific Publishing
Vol. 6, No. 1, 2020

DOI: 10.5445/KSP/1000098011/07
ISSN 2363-9881

1 Introduction

A product-line is a bundle of similar products consisting of at least two products. These products are determined by their attributes and their corresponding attribute levels. Companies offer product-lines to satisfy a higher number of consumers or consumer segments to maximize, e.g., market share or profit. Therefore, Product-line optimization is one of the most important issues in marketing science (Green and Krieger, 1985). So, new product development is key for the profitability of an enterprise. There is enormous pressure on enterprises due to globalization, shorter product life cycles and fast advances in technology (Tsafarakis et al, 2011). Around 65-79% of all new product placements flop (Hermann, 2006; Tacke et al, 2014). That is why there is a strong need to develop methods for solving product-line optimization problems because on the basis of preference measurement (especially conjoint analysis) data product line optimization is a combinatorial optimization problem and, therefore, NP-hard Kohli and Krishnamurti (1989). That means optimal solutions are very hard to find with exact methods even for middle sized issues in reasonable time (Belloni et al, 2008). A possible approach to this problem is to approximate the optimal solutions by methods from machine learning. In this work, seven different heuristics from machine learning are investigated in terms of solution quality for the probabilistic profit optimization problem.

2 The Probabilistic Profit Function

For performance evaluation of the heuristics the probabilistic profit function of Gaul et al (1995) and Gaul and Baier (2009) is used. It takes advantage of the Bradley-Terry-Luce (BTL) (Bradley and Terry, 1952; Luce, 1959) decision rule for modelling the choice of a consumer from its individual part-worths. In this context, it should be stated, that the BTL model, like the multinomial logit model, suffers from the violation of the independence of irrelevant alternatives (IIA) property. Nevertheless, in this paper a probabilistic choice rule is used, because it can lead to a more realistic consumer behavior in some situations (Gaul and Baier, 2009). In comparison to the first choice rule, one expects another structure of the solution space of the underlying optimization problem, but due to the same structure of decision variables and problem configurations no different performances of the heuristics. Another reason for using this type of

probabilistic profit function is that to this point nobody performed a product-line optimization for this model. Consumer utilities for the products are composed of a linear additive model (see Equation 1) where the utility for a customer i and a product j is the sum of the part-worths utilities of the corresponding attribute levels.

$$u_{ij} = \sum_{k=1}^{K} \sum_{l=1}^{L_k} \beta_{ikl} x_{jkl} \tag{1}$$

The probabilistic profit function maximizes the profit of new (T_1) and own established products (T_2) of a company minus the fixed costs for new products considering the individual purchase probability and a weighting factor that reflects the number of items bought by a consumer. In this context, 'own products' mean the already available products in the market of the company that performs the optimization. A description of the probabilistic profit function is given below.

$$\sum_{k=1}^{K} \sum_{l=1}^{L_k} \left(\underbrace{\sum_{j=F+E+1}^{F+E+R} x_{jkl} \left(\sum_{i=1}^{I} p_{ij} \omega_i d_{ikl} - f_{kl} \right)}_{T_1} + \underbrace{\sum_{j=F+1}^{F+E} x_{jkl} \sum_{i=1}^{I} p_{ij} \omega_i d_{ikl}}_{T_2} \right) \tag{2}$$

$\forall j, k, l \longrightarrow \max!$

subject to

$$\sum_{l=1}^{L_k} x_{jkl} \leq 1 \qquad \forall k, j = F + E + 1, ..., F + E + R \tag{3}$$

$$\sum_{l=1}^{L_k} x_{jkl} = \sum_{l=1}^{L_k+1} x_{j(k+1)l} \qquad \begin{matrix} k = 1, ..., K - 1 \\ j = F + E + 1, ..., F + E + R \end{matrix} \tag{4}$$

$$p_{ij} = \frac{\left(\sum_{k=1}^{K} \sum_{l=1}^{L_k} \beta_{ikl} x_{jkl} \right)^{\alpha}}{\sum_{j'=1}^{F+E+R} \left(\sum_{k=1}^{K} \sum_{l=1}^{L_k} \beta_{ikl} x_{jkl} \right)^{\alpha}} \qquad \forall i, j \tag{5}$$

$$x_{jkl} \in \{0,1\} \qquad \forall k,l,j = F+E+1,...,F+E+R. \tag{6}$$

Let I be the set of consumers with $i = \{1,..,I\}$, J the set of products with $j = \{1,..,J\}$ that is composed by K attributes with $k = \{1,..,K\}$ with L_k levels for each attribute with $l = \{1,...,L_k\}, \forall k \in K$. The status-quo market is made up by $j = \{1,..,F\}$ products of competing firms and $j = \{F+1,..,F+E\}$ own products. In the optimization model a number of $j = 1,...,R$ new products should be introduced into the market. So overall, there are $F+E+R$ products in the market. The part-worths β_{ikl} are the preference values of attribute k and level l for the i-th consumer. Every consumer has a certain weight $\omega_i, i = 1,...,I$, which reflects the number of products that are purchased by consumer i. The decision variable $x_{jkl}, j = F+E+1,...,F+E+R$, for the products to be introduced indicates whether a product j has attribute k with level l. The marginal profit of attribute k with level l of consumer i is denoted by d_{ikl}. To model fixed costs at the attribute levels the term f_{kl} is used. Constraint 3 ensures that each attribute k of a products j has at most one level l and Constraint 4 makes sure to offer just products with a full set of attributes and their respected levels. The BTL purchase probability p_{ij} for consumer i for buying product j is reflected by Constraint 5 whereby a calibration parameter $\alpha \geq 0$ can be used to model additional market information. Constraint 6 is the binary restriction for the decision variables x_{jkl} where $x_{jkl} = 1$ if product $j = 1,...,R$ has attribute level l of attribute k. The decision variable considers only new products to be introduced. This optimization model maximizes the marginal return of new and already established products of a firm taking fixed costs, purchase probabilities and weighting factors into account.

3 Machine Learning Techniques for Product-line Optimization

Because it is a combinatorial optimization problem, the product-line optimization is NP-hard (Kohli and Krishnamurti, 1989). Therefore, for most problem instances of a product-line optimization problem it is not possible to enumerate all product-lines for selecting the best one. There are $\binom{N}{R}$ product-lines that can be composed by selecting R products out of N permitted products. Even with highly specialized algorithms which ensure the global optimum through the use of Lagrangian relaxation in combination with branch and bound techniques,

Belloni et al (2008), Camm et al (2006) and Wang et al (2009) showed that even for middle sized problems the computation of the global optimum takes far too long for practical purposes. Hence, there is a need of gradient free optimization heuristics that give at least near optimal solutions to the underlying optimization problem. An important issue for selecting such heuristics ist the question which heuristic will perform well on the product-line optimization. A natural approach would be looking at other NP-hard combinatorial optimization problems like the travelling salesman or the knapsack problem. But as Wolpert and Macready (1997) showed with the "no free lunch theorems for search" this approach could be misleading and give wrong indications for the use of certain algorithms because there is no heuristic algorithm for optimization that is good for every problem instance. On average, heuristics are equally good over all possible problem instances. Another important issue in the field of heuristics is the massive amount of "novel" methods for solving optimization problems approximately. Soerensen (2013) stated this issue and critizes that most methods are conceptually not new and just re-combining already existing methods hiding behind other metaphors. That is why we decided to take already investigated methods or methods that are conceptually promising for product-line optimization.

Now, the heuristics used in this work will be explained shortly. Parameter configurations of the heuristics were either taken from the literature or by conducting Monte Carlo simulations. Genetic algorithms (GA) simulate the inheritance process by selection, recombination and mutation (Holland, 1975; Balakrishnan and Jacob, 1996). There are different procedures for selecting good individuals (here: product-lines) and recombining/mating them to get new offspring with new promising product-lines for the next population. The mutation operator prevents the algorithm of premature convergence and ensures a higher diversity in the population. Particle-swarm optimization (PSO) mimics the behavior of swarms of fishes or birds when foraging for food (Kennedy and Eberhart, 1995; Tsafarakis et al, 2011). Ant-colony optimization (ACO) emulates ants on their way of finding rich food sources (Dorigo and Stützle, 2004; Albritton and McMullen, 2007) where every ant represents a product-line. Good regions with rich food sources get higher selection probabilities than regions with less food. So, a solution is constructed by a probability function from one attribute to another. This probability function is updated throughout the optimization process so that attribute levels of better solutions get higher

chances to be selected in the next iteration. Simulated annealing (SA) is a technique that works on a single solution (Kirkpatrick et al, 1983; Belloni et al, 2008). It has good properties to escape from local extrema because it keeps not necessarily each solution that leads to a better objective function value. Instead, the acceptance of a solution is based on a probability which takes the difference between objective function values and a decreasing cooling schedule into account. It was one of the best algorithms in Belloni et al (2008). The multiverse optimizer (MVO) is a relativly new population-based approach (Mirjalili et al, 2015). It simulates phenomena from general relativity like black, white and worm holes. A universe is the metaphor for a solution of the optimization problem, and every variable represents an object in that universe. The population is composed of a certain number of universes. Each universe has an inflation rate which is proportional to the fitness function value of the universe. MVO follows some rules: higher inflation rates correspond to a higher probability of white holes and a lower probability of having black holes. Universes with higher inflation rates tend to send objects through a white whole to other universes and universes with lower inflation rates tend to get objects from other universes through black holes. The last rule is the existence of worm holes which send objects through worm holes all over the population of universes coming from the best universe so far. These rules are modeled with simple mathematical formulas leading to an algorithm which is capabale of finding near optimal solutions. Harmony search (HS) like GA is an evolutionary method which mimics the harmonies of music (Geem et al, 2001). These harmonies seek a best state which reflects the optimal solution of the optimization problem. To reach a best state, the algorithm follows some steps: As a population-based algorithm, HS is initialized randomly with feasible solutions, also called harmonies forming the harmony memory. After initialization, the improvisation process starts which changes the values of the variables following a harmony memory considering rate for selecting a variable from the harmony memory with a certain probability and a pitch adjusting rate which selects neighboring values at a certain probability. If a better solution is found, this solution replaces the worst solution in the harmony memory. This process is repeated until a stopping criterion is met. A memetic algorithm (MA) is a combination of a GA and a SA. The algorithm works with the same operators like a GA, but replaces the mutation operator through an annealing process with a shorter cooling schedule and lesser iterations than in classical SA. A

certain number of randomly selected solutions from the population is taken and annealed in each iteration of the GA. We should expect a better escape behavior from local optima.

4 Design of the Simulation Study

The used parameter configurations for each heuristic for the following simulation study are given by:

- GA: population size = 1000; parents rate = 0.4; mutation rate = 0.02.
- PSO: neighbourhoods = 28; particles = 18; $w_{min} = 0.1$; $w_{max} = 0.9$; $c_1 = c_2 = 2.0$; $\chi = 0.728$.
- ACO: number of ants = 500.
- SA: feature changes per temperature = 5000; cooling schedule = $\{21.0, 0.9 \cdot 21.0, ..., 0.1\}$.
- MVO: number of universes = 1000; $wep_{max} = 1.0$; $wep_{min} = 0.05$; upper bound = 1.0; lower bound = -1.0; $\alpha = 1.0$; scale = 1.0.
- HS: harmonies = 20; harmony memory considering rate = 0.99; pitch adjusting rate = 0.02.
- MA: population size = 1000; parents rate = 0.4; annealing of 10 solutions in each iteration; feature changes per temperature = 50; cooling schedule = $\{7.3, 0.9 \cdot 7.3, ..., 0.1\}$.

The values of the parameters are based on the literature and some small simulations to get good parameters for the optimization. The concrete meaning of them can be looked up in the stated literature.
To implement the constraint system into the algorithms a multinomial representation of the decision variable has been used. The main advantage is the better performance in the running time of each algorithm. To perform the calculations for the objective function these multinomial representation is converted to a binary representation. This leads to the possibility of using the fast matrix multiplication operations in *R*.

5 Simulation Study

To compare different approaches for parameter estimation and optimization in conjoint studies, in the marketing literature a long list of Monte Carlo studies is available. The main idea behind these studies is to generate a set of synthetic datasets that reflects real datasets as close as possible. Each synthetic dataset is generated according to selected factor-levels (e.g. number of respondents, number of product attributes and levels, assumed degree of heterogenity of the respondends' preference structure, assumed cost structures, assumed error variance in data collection). All synthetic datasets try to reflect the assumed variety of real datasets according to distributions of the factor-levels from former empirical studies. Then, these synthetic datasets are used to compare the different approaches and to decide which one is in which situation (factor-level-combination) the most appropriate in this contribution. The composition of the datasets is described in Baier (2014). We just refer to some of the best-known samples of such studies, e.g., (Steiner and Hruschka, 2002) and, especially, for the full factorial design used in these studies (Addelman, 1962). In our simulation study we used these comparisons for defining our Monte Carlo experiment which is close to the studies by (Carmone et al, 1978) and (Vriens et al, 1996).

There are seven heuristics evaluated in a Monte Carlo simulation. All heuristics were implemented with the *R* environment. For the implementation we built a library for the these methods. Therefore, the parameter configurations used in our experiments are taken from the literature. For the interested reader we refer to, e.g., (Balakrishnan and Jacob, 1996) for the GA implementation, (Clerc, 1999) for the PSO, (Albritton and McMullen, 2007) for the ACO, (Belloni et al, 2008) for SA, (Mirjalili et al, 2015) for MVO, and (Geem et al, 2001) for HS. The parameters used for MA are from shortened cooling schedule of (Belloni et al, 2008) for the annealing process and selection and crossover from the GA used in this article. In this simulation study the individual part-worths of the consumers are generated by a normal distribution with a mean of zero and a standard deviation of five (Tarasewich and McMullen, 2001, S. 66). The full factorial design of the Monte Carlo simulation is composed by the factors and levels from Table 1. Hence, 180 test problems are investigated (3 attributes $\times$ 3 levels for each attribute $\times$ 2 product-line sizes $\times$ 10 iterations for each test problem) which are solved by each heuristic. The number of consumers is kept

constant throughout the simulation, whereas the number of status-quo products is depending on the test problem size. For simulation purposes, the fixed costs are set to zero for each level of the attributes ($f_{kl} = 0, \forall l \in L_k, k \in K$). Furthermore, the marginal return d_{ikl} is randomly drawn from a uniform distribution from $(0, 1), \forall i \in I$. That means, every consumer has the same marginal return for each attribute level. The status quo market is composed of the status quo products, depending on the problem size, there are 5 or 20 products composing the market.

Table 1: Factors and levels for the simulation study (See also Tsafarakis et al (2011)).

Factor	Level		
attributes	3	6	9
levels per attribute	3	4	10
products per product-line	3	5	
consumers	200		
status quo products	5	20	
iterations for each problem	10		

To investigate whether the heuristics show differences in the solution quality with increasing test problem sizes, the 18 test problems are devided into three different blocks with small, middle and big sized problem instances. Therefore, the problems were ordered by the number of possible product-lines and then assigned to their respective block. The smallest number of possible solutions is $4,2 \cdot 10^4$ while the biggest problem instance has $8,3 \cdot 10^{42}$ possible solutions to the problem. In Table 2 one can find the simulation results. In the header of the table the numbers of possible solutions (product-lines) are given. Columns **best** represent the best found solution for each heuristic in ten iterations for all test problems per block divided by the best known solution of each test problem over all heuristics. E.g., if GA has a **best** value of 1.00 means, that for every test problem GA was able to find the best known solution to the six test problems of a block in at least one iteration. If a **best** value is below 1.00, a heuristic has missed the best known solution at least once. Columns **avg** show the mean of the solutions for ten iterations of all test problems per block divided by the number of test problems. If a **avg** value is 1.00, a heuristic was able to find the best known solution to the six test problems of a block in every iteration. The **time in s** column indicates the averaged run time of the heuristics for one iteration in seconds.

Table 2: Performances of algorithms for different problem sizes as share in % of the best found solution. Abbreviations: GA...genetic algorithm, PSO...particle-swarm optimization, ACO...ant-colony optimization, SA...simulated annealing, MVO...multiverse optimizer, HS...harmony search, MA...memetic algorithm.

	$4,2 \cdot 10^4$ - $3,8 \cdot 10^{10}$ solutions (small sized)			$8,3 \cdot 10^{12}$ - $1,1 \cdot 10^{22}$ solutions (middle sized)			$1,9 \cdot 10^{23}$ - $8,3 \cdot 10^{42}$ solutions (big sized)		
	best	**avg**	**time in s**	**best**	**avg**	**time in s**	**best**	**avg**	**time in s**
GA	**1.00**	**1.00**	**10.24**	**1.00**	**1.00**	**18.00**	**1.00**	0.99	51.92
PSO	**1.00**	0.99	23.99	0.99	0.99	47.08	0.94	0.89	117.53
ACO	**1.00**	0.99	42.85	0.99	0.96	57.57	0.89	0.83	126.38
SA	0.99	0.96	37.89	**1.00**	0.99	39.24	**1.00**	0.99	43.48
MVO	**1.00**	0.99	48.29	**1.00**	0.99	78.37	0.99	0.99	220.42
HS	**1.00**	0.99	22.13	**1.00**	0.99	25.85	**1.00**	0.99	**32.23**
MA	**1.00**	**1.00**	18.30	**1.00**	**1.00**	30.59	**1.00**	**1.00**	65.29

In general, the results in Table 2 show that each heuristic performed very well on small sized problems with a number of possible solutions lower or equal than $3,8 \cdot 10^{10}$. As one can see, SA missed at least one best known solution and had the worst average performance in the first block. Surprisingly, SA performed extremely well on the middle sized and big problem instances, where it is able to find every known best solution with competitive average solutions with respect to top performing heuristics like GA, HS and MA. With increasing problem sizes PSO and ACO clearly fall behind the other heuristics in terms of solution quality. Whereas the performance of PSO is acceptable for the middle sized problems, ACO gets close to the best known solution but the average performance is worse than for the other heuristics. For PSO the problem seems to be that it operates on a continuous space which after the calculations is retransformed to a discrete space. So the reason can be a loss of information throughout the learning process of the algorithm. In the current implementation ACO solely uses the consumer preferences for optimization which leads foremost to non optimal solutions for the probabilistic profit function. MA finds the best known solution to every problem instance and even gets the best known solution in every iteration.

6 Conclusion and Outlook

At first, each heuristic seems to be well suited for (near optimal) maximizing the probabilistic profit function. The best performing (with regard to solution quality) heuristic in this simulation study is the MA. But, in fact, GA, SA, MVO and HS are very close, so it is difficult to make a final conclusion which heuristic performs best in terms of solution quality. Problem instances with a higher number of possible solutions have to be investigated in the future to see more differences in the performance of the heuristics. Statistical tests have to be implemented to decide if a certain heuristic actually gives better quality solutions. For future research, problems with bigger solution spaces should be considered to get more insights in the algorithms' performances. The differences are quite small in this investigation. Furthermore, a more realistic way to model the attribute-level-structure from real world conjoint problems could be beneficial for more general-purpose statements. Another limitation of this paper is the simulation of the consumers' part-worths. More sophisticated methods would be helpful for a more realistic preference structure. Another interesting question is whether there is a difference in the performance of the algorithms when comparing different choice rules for modeling the choice behavior. A possibly fruitful approach the product-line optimization problems in particular and for combinatorial optimization in general would be the usage of quantum computing. Maybe, it is possible to solve even huge problem instances exact in reasonable time, but at least, one could implement quantum annealing which showed to have very desirable properties to escape from local extrema (Finnila et al, 1994). Another interesting method from artifical intelligence could be solving the underlying product-line optimization problem by a neural combinatorial optimization technique with reinforcement learning (Bello et al, 2016).

References

Addelman S (1962) Orthogonal main effect plans for asymmetrical factorial experiments. Technometrics 4(1):21–46. DOI: 10.1080/00401706.1962.10489985.

Albritton MD, McMullen PR (2007) Optimal product design using a colony of virtual ants. European Journal of Operational Research 176(1):498–520. DOI: 10.1016/j.ejor.2005.06.042.

Baier D (2014) Bayesian Methods for Conjoint Analysis-Based Predictions: Do We Still Need Latent Classes? German-Japanese Interchange of Data Analysis Results. Springer, Cham, pp. 103–113.

Balakrishnan PS, Jacob VS (1996) Genetic algorithms for product design. Management Science 42(8):1105–1117. DOI: 10.1287/mnsc.42.8.1105.

Bello I, Quoc V. Le H, Norouzi M, Bengio S (2016) Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940 [cs.AI]. URL: `https://arxiv.org/abs/1611.09940`.

Belloni A, Freund R, Selove M, Simester D (2008) Optimizing product line designs: Efficient methods and comparisons. Management Science 54(9):1544–1552. DOI: 10.1287/mnsc.1080.0864.

Bradley RA, Terry ME (1952) Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika 39(3/4):324–345.

Camm JD, Cochran JJ, Curry DJ, Kannan S (2006) Conjoint optimization: An exact branch-and-bound algorithm for the share-of-choice problem. Management Science 13:435–447. DOI: 10.1287/mnsc.1050.0461.

Carmone F, Green P, Jain A (1978) Robustness of conjoint analysis: Some Monte Carlo results. Journal of Marketing Research 15(2):300–303. DOI: 10.1177/002224377801500218.

Clerc M (1999) The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, Vol. 3, pp. 1951–1957. DOI: 10.1109/CEC.1999.785513.

Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press/Bradford Books, Massachusetts. ISBN: 978-0-262042-19-2.

Finnila A, Gomez M, Sebenik C, Stenson C, Doll J (1994) Quantum annealing: A new method for minimizing multidimensional functions. Chemical Physics Letters 219:343–348. DOI: 10.1016/0009-2614(94)00117-0.

Gaul W, Baier D (2009) Simulations- und Optimierungsrechnungen auf Basis der Conjointanalyse. In: Conjointanalyse: Methoden – Anwendungen – Praxisbeispiele. Springer-Verlag Berlin Heidelberg, pp. 163–182. DOI: 10.1007/978-3-642-00754-5_11.

Gaul W, Aust E, Baier D (1995) Gewinnorientierte Produktliniengestaltung unter Berücksichtigung des Kundennutzens. Zeitschrift für Betriebswirtschaftslehre 65:835–855.

Geem Z, Kim J, Loganathan G (2001) A new heuristic optimization algorithm: Harmony search. Simulation 76:60–68, Sage Publications Sage CA: Thousand Oaks, CA. DOI: 10.1177/003754970107600201.

Green PE, Krieger AM (1985) Models and heuristics for product line selection. Marketing Science 4:1–19. DOI: 10.1287/mksc.4.1.1

Hermann S (2006) Zum Messen, Managen und Monitoren der Consumer Experience. Marketingjournal 11:8–13. URL: https://www.kantartns.de/presse/pdf/autorenbeitraege/200611marketingjournal_tns_infratest_herrmann.pdf.

Holland JH (1975) Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. The MIT Press. DOI: 10.1086/418447.

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks IV, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

Kirkpatrick S, Gelatt Jr CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680, American association for the advancement of science. DOI: 10.1126/science.220.4598.671.

Kohli R, Krishnamurti R (1989) Optimal product design using conjoint analysis: Computational complexity and algorithms. Journal of Operational Research 40:186–195. DOI: 10.1016/0377-2217(89)90329-9.

Luce RD (1959) Individual Choice Behavior: A Theoretical Analysis. John Wiley.

Mirjalili S, Mirjalili S, Hatamlou A (2015) Multi-verse optimizer: A nature-inspired algorithm for global optimization. Neural Computing and Applications, pp. 1–19, Springer London. DOI: 10.1007/s00521-015-1870-7.

Soerensen K (2013) Metaheuristics – the metaphor exposed. International Transactions in Operational Research In Press. DOI: 10.1111/itor.12001.

Steiner W, Hruschka H (2002) Produktliniengestaltung mit genetischen Algorithmen. Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung 54(7):575–601. DOI: 10.1007/BF03372688.

Tacke G, Vidal D, Haemer J (2014) Profitable innovation. Simon-Kucher & Partners Strategy & Marketing Consultants GmbH. ISBN: 978-3-981681-22-2.

Tarasewich P, McMullen PR (2001) A pruning heuristic for use with multisource product design. European Journal of Operational Research 128:58–73, Elsevier Science B.V., Radarweg 29, 1043 NX Amsterdam, The Netherlands. DOI: 10.1016/S0377-2217(99)00350-1.

Tsafarakis S, Marinakis Y, Matsatsinis N (2011) Particle swarm optimization for optimal product line design. International Journal of Research in Marketing 28(1):13–22. DOI: 10.1016/j.ijresmar.2010.05.002.

Vriens M, Wedel M, Wilms T (1996) Metric conjoint segmentation methods: A Monte Carlo comparison. Journal of Marketing Research 33(1):73–85. DOI: 10.2307/

3152014.
Wang X, Camm JD, Curry DJ (2009) A branch-and-price approach to the share-of-choice product line design problem. Management Science 55:1718–1728. DOI: 10.1287/mnsc.1090.1058.
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1):67–82. DOI: 10.1109/4235.585893.