A Comparison of the Taguchi Method and Evolutionary Optimization in Multivariate Testing

Jingbo Jiang,[1] Diego Legrand,[1] Robert Severn,[1] Risto Miikkulainen[1,2]
[1]Sentient Technologies, Inc.
One California, California St Suite 2300
San Francisco, CA 94111
[2]The University of Texas at Austin
2317 Speedway, Stop D9500
Austin, TX 78712

arXiv:1808.08347v1 [cs.NE] 25 Aug 2018

Abstract

Multivariate testing has recently emerged as a promising technique in web interface design. In contrast to the standard A/B testing, multivariate approach aims at evaluating a large number of values in a few key variables systematically. The Taguchi method is a practical implementation of this idea, focusing on orthogonal combinations of values. This paper evaluates an alternative method: population-based search, i.e. evolutionary optimization. Its performance is compared to that of the Taguchi method in several simulated conditions, including an orthogonal one designed to favor the Taguchi method, and two realistic conditions with dependences between variables. Evolutionary optimization is found to perform significantly better especially in the realistic conditions, suggesting that it forms a good approach for web interface design in the future.

Keywords: Evolution algorithm, Taguchi method, multivariate testing, web interface design

Introduction

Large-scale testing is widely regarded as key to success in web interface design (Moe and Fader 2004; Kohavi and Thomke 2017), but such testing is difficult to implement in practice. The number of potential variations grows very fast as a function of possible changes to the page, and there is rarely enough traffic available to test them comprehensively.

Current methods for optimizing web interfaces are thus based on limited testing. They include A/B testing, full factorial multivariate testing, partial factorial multivariate testing (the Taguchi method), and other statistical techniques. These methods are broadly used (Dreze and Zufryden 1997; Taguchi and Rajesh 2000; Dixon, Enos, and Brodmerkle 2011), but they are limited in terms of how many changes can be tested, and/or they assume that the changes have independent effects.

This paper evaluates an alternative approach based on population-based search. Because such search is based on intelligent sampling of the entire space, instead of statistical modeling, it can potentially overcome those shortcomings. Crossover and mutation can traverse massive solution spaces efficiently, discovering dependencies and using them as building blocks (Goldberg 2006; Floreano, Drr, and Mattiussi 2008; Deb and Myburgh 2016; Shahrzad, Fink, and Miikkulainen 2018). It can therefore search the space in a more comprehensive manner, and effectively find solutions that the other methods miss.

Evolution is implemented as the search method in Sentient Ascend web-interface optimization system (Miikkulainen et al. 2017; Miikkulainen et al. 2018). This paper compares the Ascend implementation to the state-of-the-art statistical method of Taguchi optimization in simulated experiments. The results show that evolution is indeed more powerful optimizer, especially under realistic conditions where there are nonlinear dependencies between variables. It is therefore a promising foundation for designing optimization applications in the future, and increases the potential for more powerful AI applications in related fields.

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The Taguchi Method

Ideally, the best web interface design would be decided based on full factorial multivariate testing. That is, each possible combination of N variables with K values (or levels) would be implemented as a candidate. For example, a variable might be the color or the position of a button; the levels would then be the possible colors and positions. A full factorial analysis would require testing all K^N combinations, which is prohibitive in most cases.

Instead, the Taguchi method specifies a small subset of these combinations to test using orthogonal arrays. An Taguchi orthogonal array is an matrix where each column corresponds to a variable and each row to a candidate to test. Each value represents the level setting for a given variable and experiment. It has the following properties:

- The dot product between any two normalized column vectors is zero.
- For every variable column, each level appears the same amount of times

There are multiple ways of creating orthogonal arrays (Brouwer, Cohen, and Nguyen 2006; Hedayat, Sloane, and Stufken 2018). Table 1 shows an example of an orthogonal array of nine combinations, resulting from testing four variables of three levels each.

Candidate	Var 1	Var 2	Var 3	Var 4	Performance
1	0	0	0	0	p1
2	0	1	2	1	p2
3	0	2	1	2	p3
4	1	0	2	2	p4
5	1	1	1	0	p5
6	1	2	0	1	p6
7	2	0	1	1	p7
8	2	1	0	2	p8
9	2	2	2	0	p9

Table 1: Example Taguchi array of four variables with three levels each

To compute the effect of a specific variable level, we average the performance scores of the candidates corresponding to combinations for that level setting. Because in an orthogonal array, all levels of the other variables are tested an equal amount of times, their effects cancel out, assuming each variable is independent(Hedayat, Sloane, and Stufken 2018). For example, to compute the effect of level 2 of variable 3 in table1, we average the scores of candidates 2, 4 and 9. Similarly, for level 1, we average the scores of candidates 3, 5 and 7.

In a Taguchi experiment, all the candidates (rows) in the orthogonal table are tested, and the scores for candidates that share the same level for each variable are averaged in this manner. We can then predict a best performing combination by selecting, for each variable, the level with the best such average score.

The Taguchi method is a practical approximation of factorial testing. However, the averaging steps assume that the effects of each variable are independent, which may or may not hold in real-world experiments. In contrast, population-based search makes no such assumptions, as will be discussed next.

Evolutionary Optimization

Evolution optimization is a broadly used method for combinatorial problems, building on population-based search. It has certain advantages compared to diagnostic methods, including fewer restrictions on input variables, robustness to environmental changes, good scale-up to large and high-dimensional spaces, and robustness to deceptive search spaces and nonlinear interactions (Branke 2012). The main idea is that instead of constructing the winning combination though independence assumptions (as in Taguchi), the winner is searched for using crossover and mutation operators.

The evolution algorithm used in this paper is that of Sentient Ascend, a conversion optimization product for web interfaces (Miikkulainen et al. 2017). The basic unit of the method is the candidate's genome, which is a list representing the elements and values of the web interface. For example, the genome **[2,4,5,3]** defines a web page with four changeable parts, i.e. genes, with 2, 4, 5, and 3 different choices each. The choices are represented as one-hot vectors, and are concatenated to form the genome. The control candidate is the genome representing default web settings, consisting of the first dimension in each vector:

$$[[1,0],[1,0,0,0],[1,0,0,0,0],[1,0,0]]$$

The candidates in the first generation consist of all genomes that are one gene different from the control. Thus, the number of candidates in this generation is the sum, over all genes, of the number of levels minus one, i.e. $1+3+4+2=10$ in this example. In all future generations, the total number of candidates stays the same; a certain percentage of candidates (e.g. 20%) are chosen as elites, staying on to the next generation. The remaining (e.g. 80%) candidates are formed by crossover from those elites (Miikkulainen and Long 2017). The encoding of each candidate also has a chance to mutate (i.e. specify a different choice for each part), according to probabilities specified in the parameter setting. The evolution process ends after a prespecified number of generations or after a suitable candidate is found.

After an evolutionary simulation, a prior estimate of the conversion rate is obtained as the average of all candidates tested. A probability to beat control is computed for each candidate based on this prior and its individual estimate, and the one with the highest probability is selected as the winner.

Evaluation in a Simulator

The performance of the Taguchi method and Evolutionary optimization was measured in simulated experiments of designing web interfaces for maximum conversion rate. In the simulation, an evaluator is first constructed to calculate a candidate's true conversion rate based on the values it specifies for each variable. Simulated traffic is then distributed to candidates and conversions are assigned probabilistically based on candidates' true conversion rate. The observed conversion rates are then used as the scores of the candidates in Taguchi and evolution methods.

CR	true conversion rate
c	candidate
n	the number of variables
W^0	bias (i.e. CR of the control candidate)
$W_i^1(c)$	impact of the candidate c's value for variable i
$W_{jk}^2(c)$	interaction between candidate c's values of variables j and k

Table 2: Evaluator Denotation

By setting the parameters in Table 2, different kinds of evaluators can be defined. The conversion rate of simple linear evaluator is based on only bias and weight for each individual variable:

$$CR[c] = W^0 + \sum_{i=1}^{n} W_i^1(c). \quad (1)$$

The bias represents the conversion rate of the control candidate; the different choices for each variable add or

subtract from the control rate. A non-linear evaluator is designed to include interactions between variables:

$$CR[c] = W^0 + \sum_{i=1}^{n} W_i^1(c) + \sum_{j=1}^{n} \sum_{k=j+1}^{n} W_{j,k}^2(c). \quad (2)$$

In addition to bias and individual variable contributions, it includes contributions for each pair of variables.

Both the Taguchi candidates and the evolution candidates are represented in the same way, as concatenations of one-hot vectors representing the levels for each variable in the Taguchi method, and actions for each gene in evolution. The total traffic for the Taguchi method and evolution algorithm is set to be equal, distributed evenly to all Taguchi candidates, but differently for evolution candidates based on how many generations they survive.

Table 3 specifies the parameter settings used in all experiments with evolution, and the evaluator bias rate.

Parameter	Value
Number of generations	8
Mutation weight	0.01
Elite percentage	20%
Evaluator bias (i.e. control's conversion rate) W^0	0.05

Table 3: Default parameter setting

Experimental Results

Three experiments were run comparing the Taguchi method with evolutionary optimization: two experiments where the variables had independent effect, one with a uniform and the other with varied number of levels; and one experiment with dependencies between pairs of variables. In addition to comparing the ability of these methods to find good candidates as the final result of the experiment, their performance during the experiment was also compared. The result curves demonstrate statistical means and 95% credible intervals of 20 repeated experiments, under the same example settings in each section.

Independent Variables with Uniform Levels

The Taguchi method assumes that the variables are independent. The first experiment was designed accordingly: It uses a linear evaluator that assumes all changes are independent, and a simple genome that results in few rows in the Taguchi array. These are the ideal conditions for the Taguchi method, and it is expected to perform well. The best settings for the Taguchi method are those with uniform numbers of levels across all variables (Adobe 2017):

Setting 1: Three variables with two levels each, i.e. **[2, 2, 2]**, with $2^3 = 8$ combinations, resulting in four rows;

Setting 2: Four variables with three levels each, i.e. **[3, 3, 3, 3]**, with $3^4 = 81$ combinations, resulting in nine rows; and

Setting 3: Five variables with four levels each, i.e. **[4, 4, 4, 4, 4]**, with $4^5 = 1024$ combinations, resulting in 16 rows.

The Taguchi arrays for these settings can be found in orthogonal array libraries (Kuhfeld 2018). The learning curves under all three settings are similar, so Setting 2 will be used as an example.

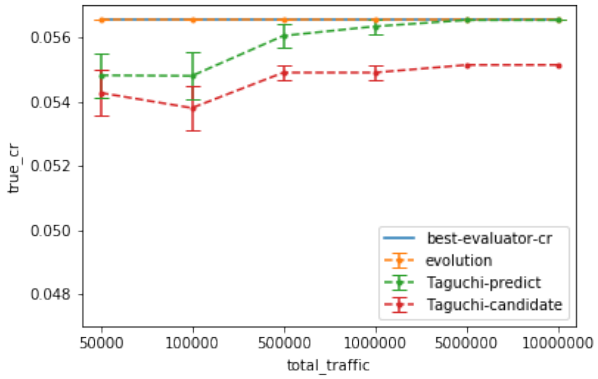

Figure 1: **True conversion rate performance on the [3,3,3,3] setting for the Taguchi method and evolution with increasing amount of traffic. The *evolution* line is the best candidate chosen by evolution algorithm; the *Taguchi-predict* line is the combined candidate from Taguchi variable analysis; *Taguchi-candidate* is the highest scored candidate in original input Taguchi array. Evolution algorithm performs significantly better with small amount of traffic; after about 500,000, both methods perform similarly.**

Figure 1 shows the true conversion rates of the best candidates under Setting 2 with increasing traffic. The true conversion rate for the best evolution candidate is steady and high at all traffic levels. The best predicted Taguchi candidates true conversion rate lags behind evolution with low traffic, but eventually catches up as traffic increases. The best tested Taguchi candidate remains significantly below both curves. Thus, under ideal conditions for Taguchi, both methods find equally good solutions given enough traffic (i.e. more than 500,000). With low traffic, the best evolutionary approach performs significantly better.

Independent Variables with Variable Levels

In real world applications, such as optimization of commercial websites, the design space may be rather complex; in particular, the number of levels for each variable is not likely to be the same. In the second experiment, while still maintaining independence between variables, the genome structure is changed to:

[3, 6, 2, 3, 6, 2, 2, 6],

i.e. three variables with two level each, two variables with three level each, and three variables with six level each. In this setting with 15,552 combinations, the Taguchi array needs 36 rows (Kuhfeld 2018).

The result in Figure 2 shows that with a more complex problem, both evolution and Taguchi require more traffic in order to find good solutions. However, evolution produces

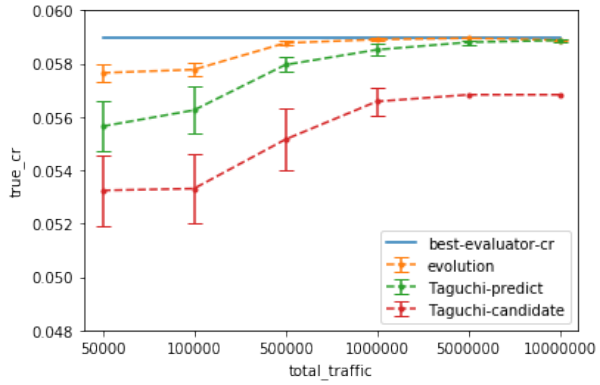

Figure 2: **True conversion rate performance with a more complex genome. Both methods take longer to find good candidates; Taguchi is now comparable to evolution only with the highest amounts of traffic.**

significantly better candidates than Taguchi at almost all traffic levels: The two methods are comparable only for very high traffic, i.e. greater than 5,000,000. The prediction process of Taguchi still provides a major advantage beyond its input set.

Interactions Between Variables

Another important challenge in real-world applications is that the variables are not likely to be independent. For example, text color and background color may interact—for instance, blue text on a blue background would perform poorly compared to blue text on a white background. The nonlinear evaluator is designed to test the ability of the two methods to handle this kind of interactions. The example uses genome in Section **Independent Variables with Variable Levels**.

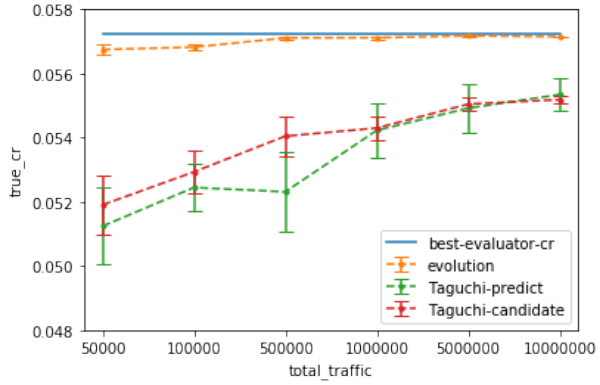

Figure 3: **True conversion rate performance with interacting variables. Evolutionary optimization now results in significantly better candidates at all traffic levels. The *Taguchi-predict* result is similar to *Taguchi-candidate*, suggesting that the interactions render the construction process ineffective.**

Figure 3 shows that when the independence assumption for Taguchi method is broken, the best predicted Taguchi candidate's true conversion rate is no longer comparable with evolution's. Furthermore, its predicted best candidate does not even significantly outperform its best tested candidate. Interestingly, the performance of the evolutionary algorithm is not significanly worse with interacting vs. independent variables, demonstrating its ability to adapt to complicated real-world circumstances.

Performance During Experiment

The main goal in conversion optimization is to find good candidates that can be deployed after the experiment. However, in many cases it is also important to not decrease the site's performance much during the experiment. Evolution continuously creates improved candidates as it learns more about the system, whereas the Taguchi method generates a single set of candidates for the entire test—it therefore provides continual improvement on the site even during the experiment.

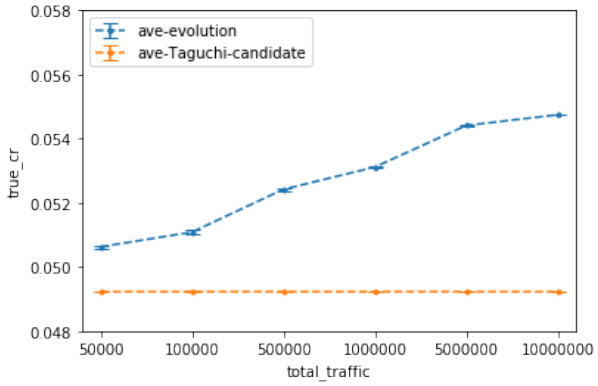

Figure 4: **Average true conversion rate of candidates with evolution and Taguchi methods during the experiment. While Taguchi candidates do not change, evolution continuously comes up with better candidates, thus increasing performance during the experiment. It therefore forms a good approach for campaigns with fixed duration as well.**

This principle is illustrated graphically in Figure 4, using the linear evaluator and genome from Section **Independent Variables with Variable Levels** as the example setting. The Taguchi's candidates average performance stays the same throughout the increasing traffic, whereas evolution's candidates perform, on average, better with more traffic, i.e. while the experiment progresses. It therefore forms a good approach in domains where performance matters during the experiment, in particular in campaigns that run only for a limited duration.

Discussion and Future Work

The two methods tested in the experiments of this paper, Taguchi and evolution, are both beneficial in web interface design. Taguchi's appeal is its high reduction of rows

compared to full factorial combinations, which works best when the genome structure is rather simple. When the genome becomes larger and more complex, its performance falls behind that of evolution. Most importantly, if there are nonlinear interactions between the variables, the method cannot keep up with them: the best-candidate construction does not improve upon its initial best candidates, and it performs much worse than evolution at all traffic levels.

In contrast, the process of searching for good candidates in evolution is based on crossover and mutation, and therefore is not affected much by interactions. Evolution discovers good combinations, and constructs future candidates using them as building blocks. As long as the interactions occur within the building blocks, they will be included and utilized the same way as independent contributions. Given how common interactions are in real-world problems, this ability should turn out important in applying optimization to web interface design in the future, enabling more powerful AI applications in related areas.

Conclusion

This paper demonstrates that (1) even with ideal conditions, the Taguchi method does not exceed performance of evolution, and (2) with low traffic in ideal conditions, evolution performs significantly better. (3) As the experiment configuration becomes more complex, Taguchi requires more traffic to match evolution's performance. (4) With nonlinear interactions between variables, Taguchi's construction process breaks down, and it no longer improves upon best initial candidates. (5) In contrast, evolution is able to find good candidates even with nonlinear interactions. Furthermore, (6) evolution improves during the duration of the experiment, making it a good choice for campaigns as well. Evolutionary optimization is thus a superior technique for improving conversion rates in web interface design.

References

[Adobe 2017] Adobe. 2017. *Best Practices for a Multivariate Test*. https://marketing.adobe.com/resources/help/en_US/tnt/help/r_Best_Practices_for_a_Multivariate_Test.html, retrieved on 8/24/2018.

[Branke 2012] Branke, J. 2012. *Evolutionary optimization in dynamic environments*, volume Vol. 3. Springer Science & Business Media.

[Brouwer, Cohen, and Nguyen 2006] Brouwer, A. E.; Cohen, A. M.; and Nguyen, M. V. 2006. Orthogonal arrays of strength 3 and small run sizes. *Journal of Statistical Planning and Inference* 136(9):3268–3280.

[Deb and Myburgh 2016] Deb, K., and Myburgh, C. 2016. Breaking the billion-variable barrier in real-world optimization using a customized evolutionary algorithm. *In Proceedings of the Genetic and Evolutionary Computation Conference* pp. 653–660. ACM.

[Dixon, Enos, and Brodmerkle 2011] Dixon, E.; Enos, E.; and Brodmerkle, S. 2011. A/B testing of a webpage. *U.S. Patent 7,975,000*. Washington, DC: U.S. Patent and Trademark Office.

[Dreze and Zufryden 1997] Dreze, X., and Zufryden, F. 1997. Testing web site design and promotional content. *Journal of Advertising Research* 37(2):77–91.

[Floreano, Drr, and Mattiussi 2008] Floreano, D.; Drr, P.; and Mattiussi, C. 2008. Neuroevolution: from architectures to learning. *Evolutionary Intelligence* 1(1):47–62.

[Goldberg 2006] Goldberg, D. E. 2006. *Genetic algorithms*. Pearson Education India.

[Hedayat, Sloane, and Stufken 2018] Hedayat, A. S.; Sloane, N. J. A.; and Stufken, J. 2018. *Orthogonal arrays: theory and applications*. Springer Science & Business Media.

[Kohavi and Thomke 2017] Kohavi, R., and Thomke, S. 2017. The surprising power of online experiments. *Harvard Business Review* 95(5):pp. 7482.

[Kuhfeld 2018] Kuhfeld, W. F. 2018. Statistical Analysis System. https://support.sas.com/techsup/technote/ts723.html.

[Miikkulainen and Long 2017] Miikkulainen, Risto, H. S. N. D., and Long, P. 2017. How to select a winner in evolutionary optimization? *In Computational Intelligence (SSCI), 2017 IEEE Symposium Series on,* pp. 1–6. IEEE.

[Miikkulainen et al. 2017] Miikkulainen, R.; Iscoe, N.; Shagrin, A.; Cordell, R.; Nazari, S.; Schoolland, C.; Brundage, M.; Epstein, J.; Dean, R.; and Lamba, G. 2017. Conversion rate optimization through evolutionary computation. *In Proceedings of the Genetic and Evolutionary Computation Conference* pp. 1193–1199. ACM.

[Miikkulainen et al. 2018] Miikkulainen, R.; Iscoe, N.; Shagrin, A.; Rapp, R.; Nazari, S.; McGrath, P.; and et al, C. S. 2018. Sentient Ascend: AI-based massively multivariate conversion rate optimization. *In Proceedings of the Thirtieth Innovative Applications of Artificial Intelligence Conference. AAAI*.

[Moe and Fader 2004] Moe, W. W., and Fader, P. S. 2004. Dynamic conversion behavior at e-commerce sites. *Management Science* 50(3):326–335.

[Shahrzad, Fink, and Miikkulainen 2018] Shahrzad, H.; Fink, D.; and Miikkulainen, R. 2018. Enhanced optimization with composite objectives and novelty selection. *In Proceedings of the 2018 Conference on Artificial Life*.

[Taguchi and Rajesh 2000] Taguchi, G., and Rajesh, J. 2000. New trends in multivariate diagnosis. *Sankhy: The Indian Journal of Statistics, Series B* 233–248.