Sawtooth Software

RESEARCH PAPER SERIES

Multi-Objective Searches in Conjoint Analysis: An Intuitive Explanation

Bryan Orme
Sawtooth Software, Inc.

3210 N. Canyon Rd., Provo, Utah
+1 801 477 4700
www.sawtoothsoftware.com

Multi-Objective Searches in Conjoint Analysis: An Intuitive Explanation

Bryan Orme, Sawtooth Software

(We assume the reader is already familiar with market simulators for conjoint analysis and product search optimization. For an introduction to market simulators, see: http://www.sawtoothsoftware.com/download/techpap/introsim.pdf. For an introduction to search procedures for product optimization, see: http://www.sawtoothsoftware.com/download/techpap/asmtech.pdf).

Executive Summary

In conjoint/choice analysis, we often build choice simulators to investigate product or product line solutions that maximize some criterion such as share, revenue, or profit. However, solutions that maximize relative share often yield low profit. Conversely, profit-maximizing solutions can give up a great deal in terms of relative share. Multi-objective searches find solutions that optimize the trade-off among multiple objectives like share and profit, allowing organizations to focus on solutions that can satisfy multiple goals. As an example, solutions can be identified that nearly optimize profit while also nearly optimizing relative share of preference.

Introduction

Search-enabled choice simulators for conjoint/choice analysis can find solutions that maximize criteria such as share, revenue, utility, or profit. If the search space is small enough, exhaustive search can examine all possible product configurations and ensure the globally-optimal solution. When the search space becomes too large, we can use heuristic algorithms such as hill-climbing (called "Grid Search" within our software) or genetic algorithms. With heuristic search algorithms, finding the globally-optimal solution is no longer guaranteed, but it is still often found.

Sawtooth Software released its first advanced market (choice) simulation module for optimization searches in 2003. That software searched based on a *single* criterion at a time: utility, purchase likelihood, share, revenue, profit, or cost. Scott Ferguson and Garrett Foster of North Carolina State University presented a paper at the 2013 Sawtooth Software Conference that explained the benefits of multi-objective search. They described how taking into consideration multiple optimization criteria (such as *both* share and profit) can lead to even better solutions that improve the likelihood of success for the organization. Both exhaustive and genetic algorithms can be employed for multi-objective search. Although Ferguson and Foster were not the first to propose and describe multi-objective genetic algorithms (MOGAs), they were the first to present this at the Sawtooth Software conferences.

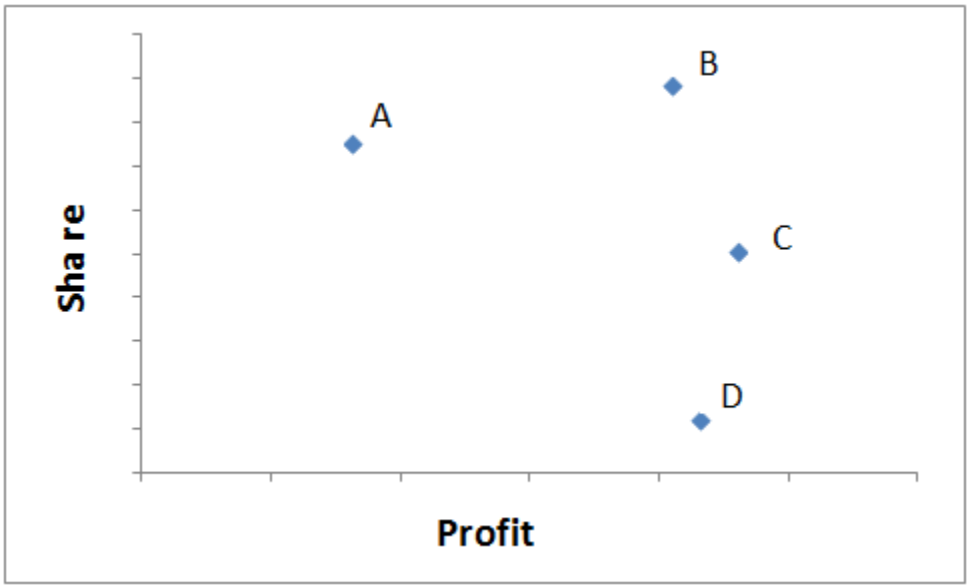
According to Ferguson and Foster, there is a rich literature in the engineering field regarding multi-objective search spanning at least 25 years.

In this paper, we describe in an intuitive way how multi-objective search works, first for the exhaustive search case and then for the genetic algorithm case. Multi-objective search is now available in Sawtooth Software's Choice Simulator, integrated within Lighthouse Studio or also available as a standalone version.

Multiple Objectives and the Concept of Dominance

For ease of illustration, let's imagine that we have run an optimization for a conjoint analysis (CBC) problem that has only four possible product configuration solutions. For this optimization, we knew cost information for certain attribute levels such that we could compute not only the share of preference for each product concept but also the relative revenue (computed as share of preference times price) and relative profit (computed as revenue less the cost). In Figure 1, we plot the relative Profit on the X axis and the Share of Preference on the Y axis for the four different solutions (each of which has been simulated in competition with other fixed competitive offerings along with the None alternative).

Figure 1

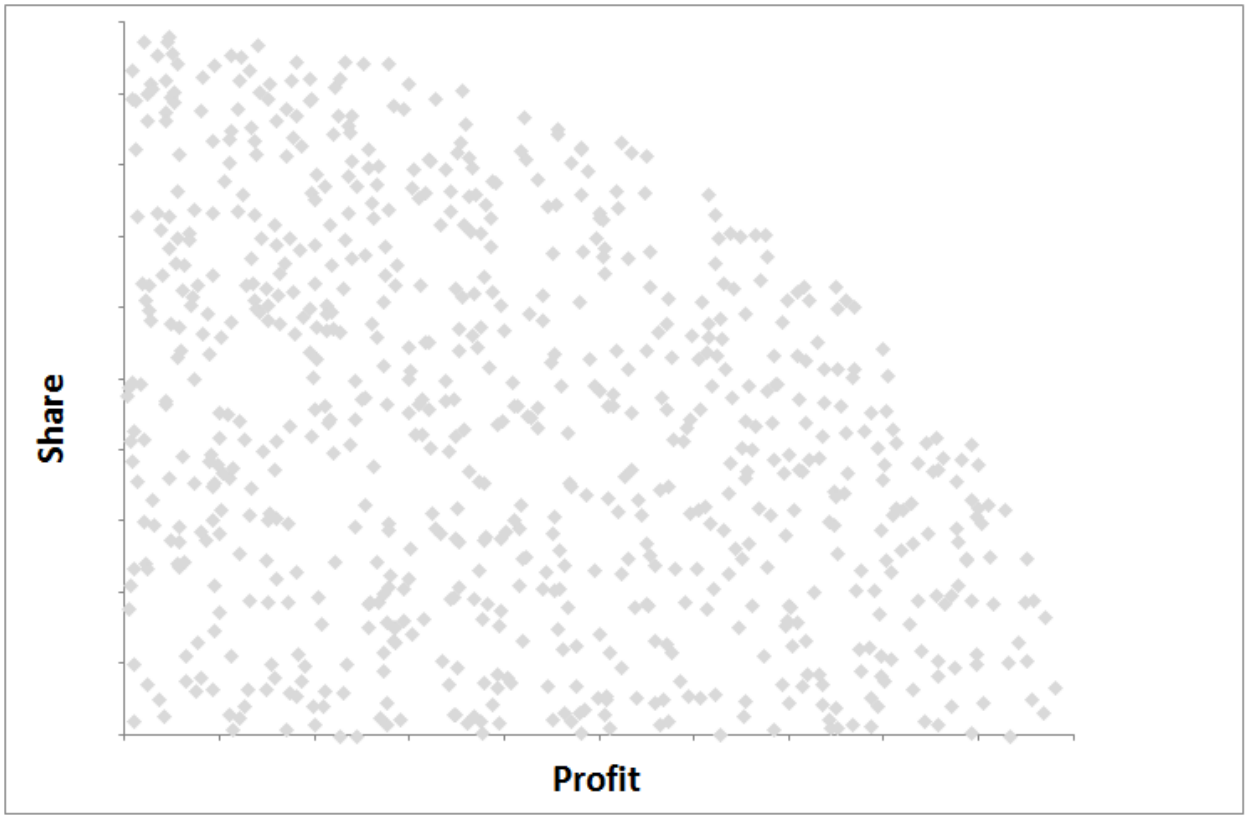


In Figure 1, solution B has both higher share and higher profit than solution A. Thus, A is dominated by solution B on both dimensions. Solution C has higher profit than solution B, but lower share. Neither B nor C is dominated (or equaled) on *both* dimensions by any other solution. D, however, is dominated by C on both dimensions. Thus, the Pareto efficient frontier of non-dominated solutions is defined by the set {B, C}. We call solutions B and C rank order 1 solutions, because they are dominated by no other solutions. The goal of multi-objective search is to find a large number of diverse rank order 1 solutions that are as distant from the origin as possible.

Searching based on multiple objectives often makes a great deal of sense for organizations. A solution that maximizes profits (such as C in Figure 1) is probably not as compelling as one (such as B) that obtains almost the same predicted profits but results in *much* larger share of preference.

Now that we've introduced the notion of dominance, efficient frontier, and rank order 1 solutions, let's consider a larger problem: a search space with 750 possible solutions, plotted in Figure 2 in terms of share and profit.

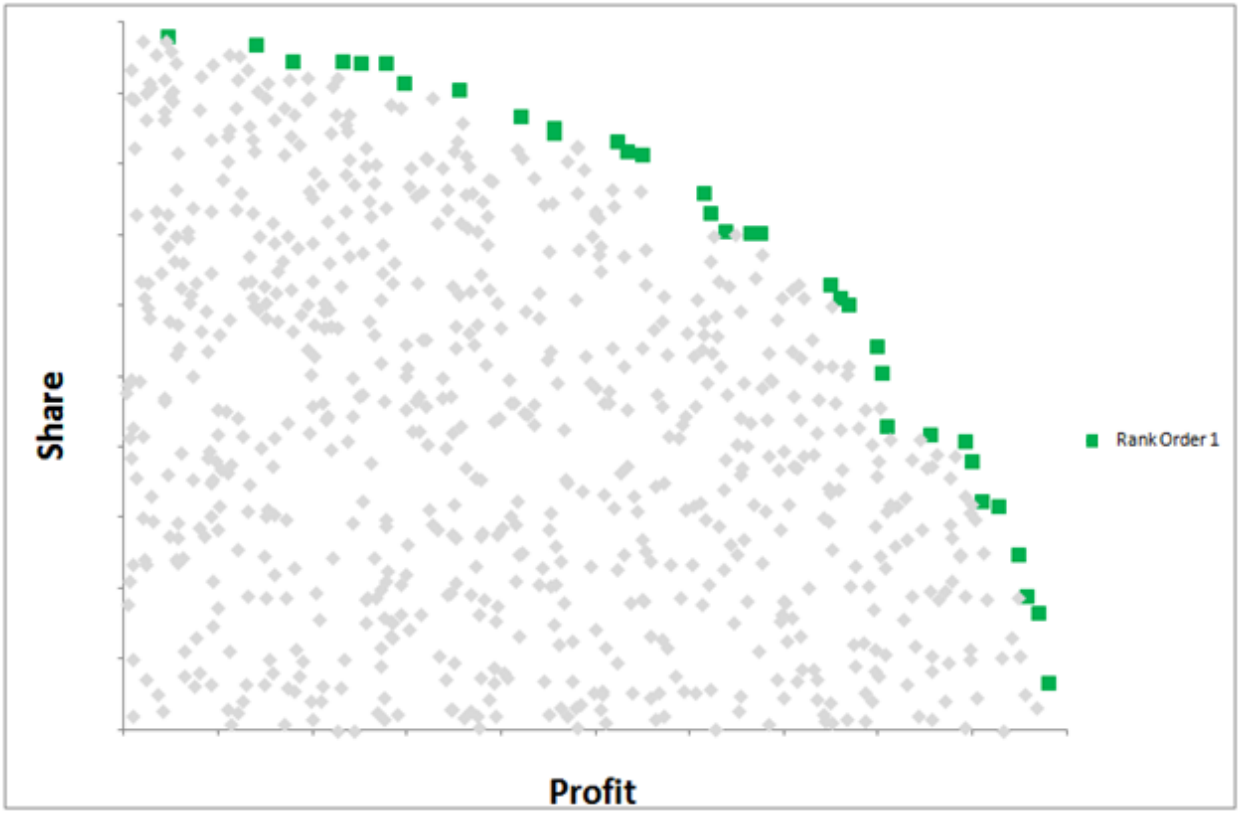
Figure 2



In practice, the solutions may not be as evenly spaced across the response surface or uniformly convex as shown in this contrived example.

Using the dominance criterion, we can determine which among the 750 solutions are not completely dominated on *both* dimensions by any other solution (or equaled on one dimension while being dominated on the other). These are the rank order 1 solutions indicated by green markers along the efficient frontier in Figure 3.

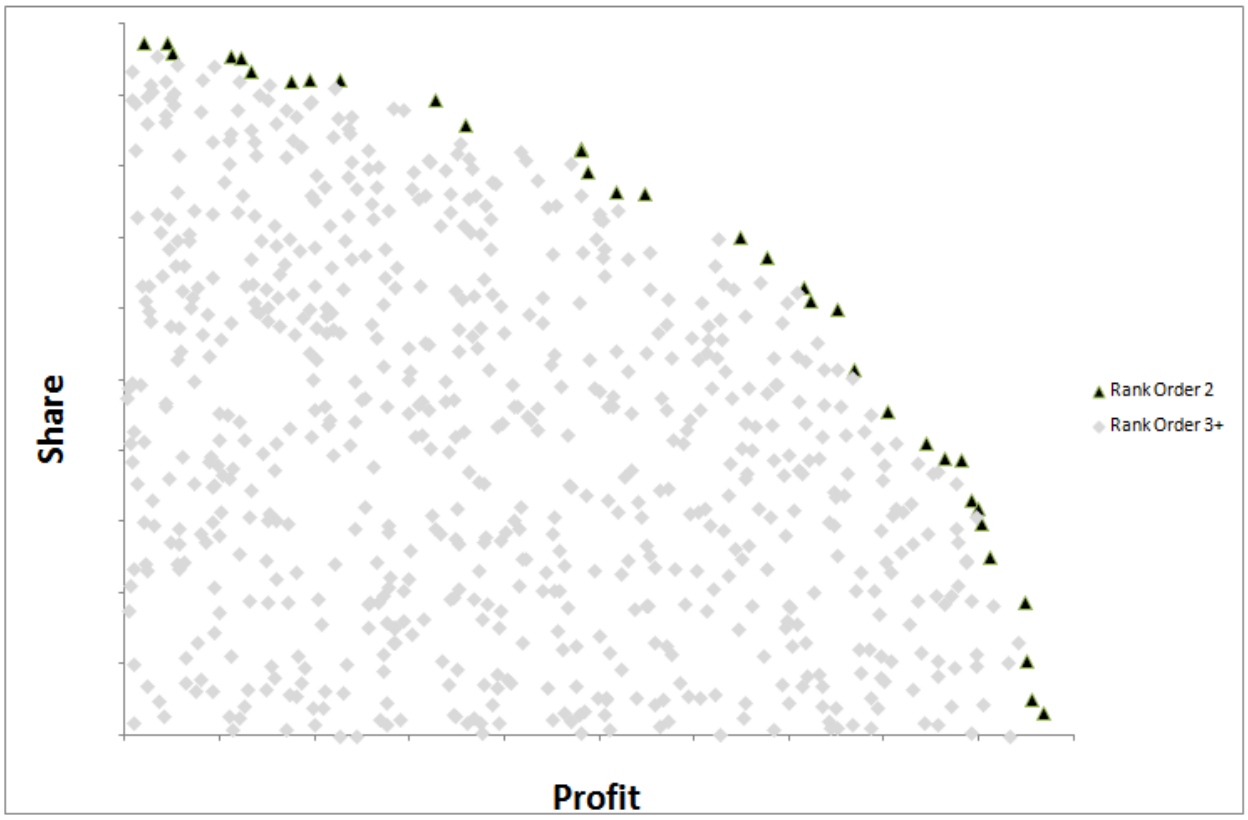
Figure 3



The rank order 1 solutions are probably the most useful for the researcher to consider. Which rank order 1 solution is best depends on the organization's goals, the technical feasibility of each solution, and perhaps political factors within the organization. But there are many other solutions nearly as good as these rank order 1 solutions (e.g. the rank order 2 solutions shown further below). If for some reason the rank order 1 solutions are not palatable to the organization, the researcher may want to examine additional near-optimal solutions very close to the efficient frontier.

To find rank order 2 solutions, we (temporarily) delete the rank order 1 solutions from consideration and then determine which of the remaining solutions are not dominated by any other solution on *both* criteria. These rank order 2 solutions are displayed in Figure 4 as black triangles.

Figure 4



We can continue the same procedure to find the rank order 3, etc., solutions. We (temporarily) delete all rank order 1 and 2 solutions and then determine which remaining solutions are not dominated, leading to rank order 3 solutions marked in Figure 5 as blue circles.

Figure 5

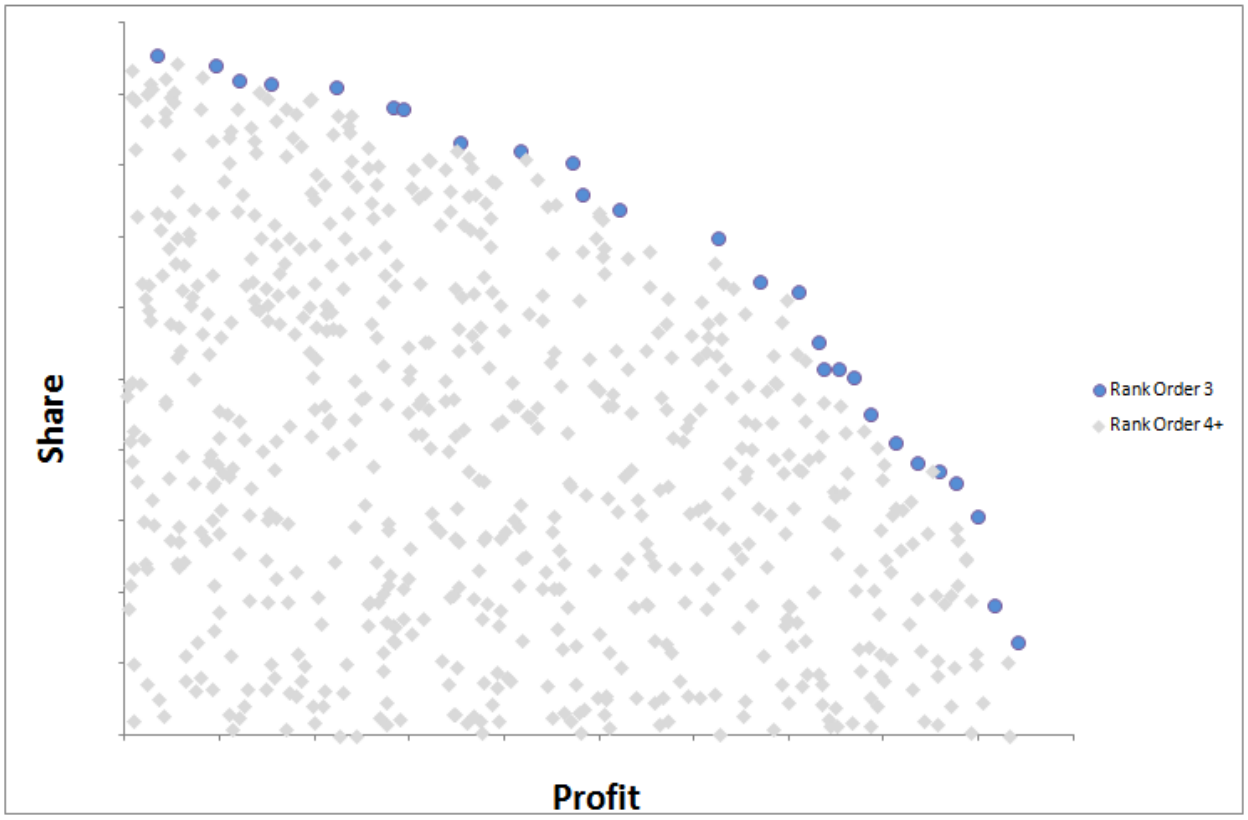
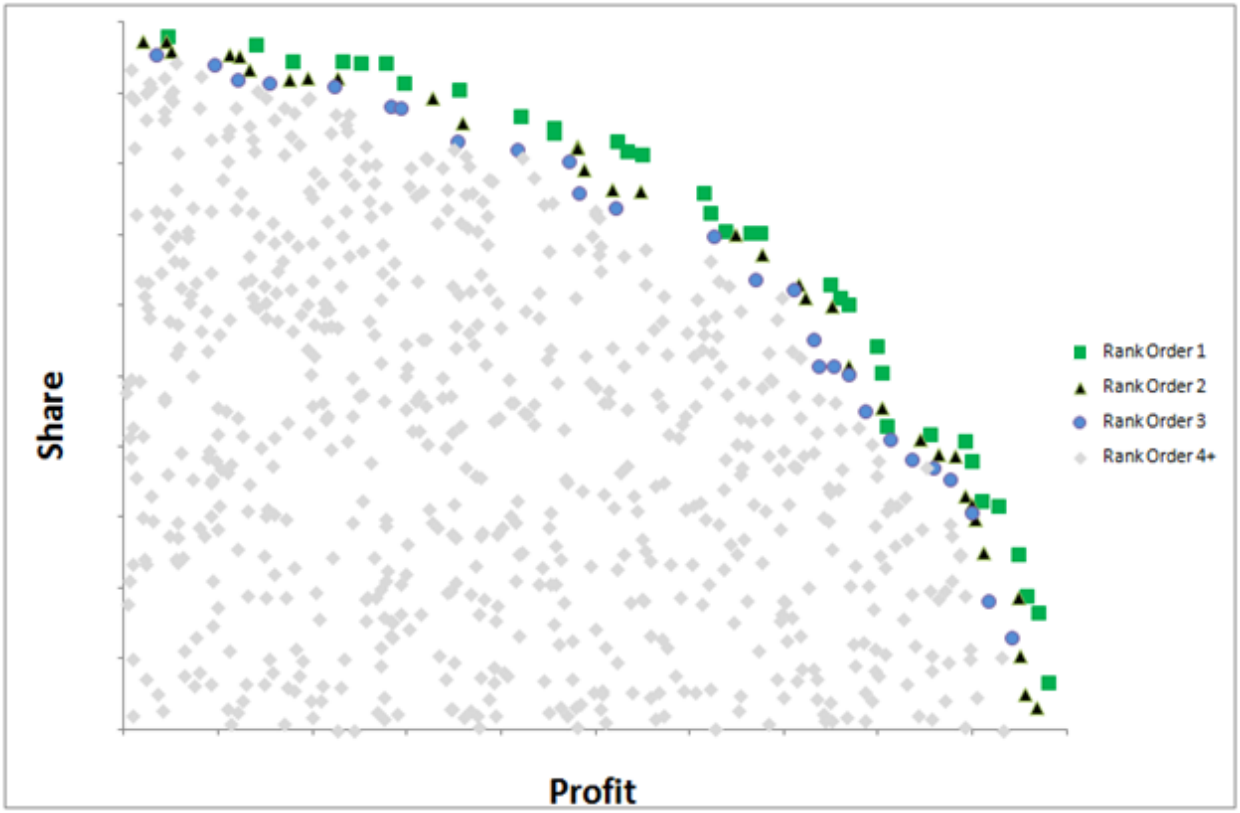


Figure 6 displays all 750 solutions again, with markers indicating rank order 1, 2, and 3 solutions.

Figure 6



We've described the algorithm for determining rank order of domination, but we've found that it is very computationally intensive to compute rank order in this way when the number of solutions exceeds about 5000. Therefore, when the number of solutions to sort by rank order of domination exceeds 5000, our software defaults to an extremely fast bounded best order sort (BBOS) algorithm that produces extremely similar results (Roy et al. 2018).

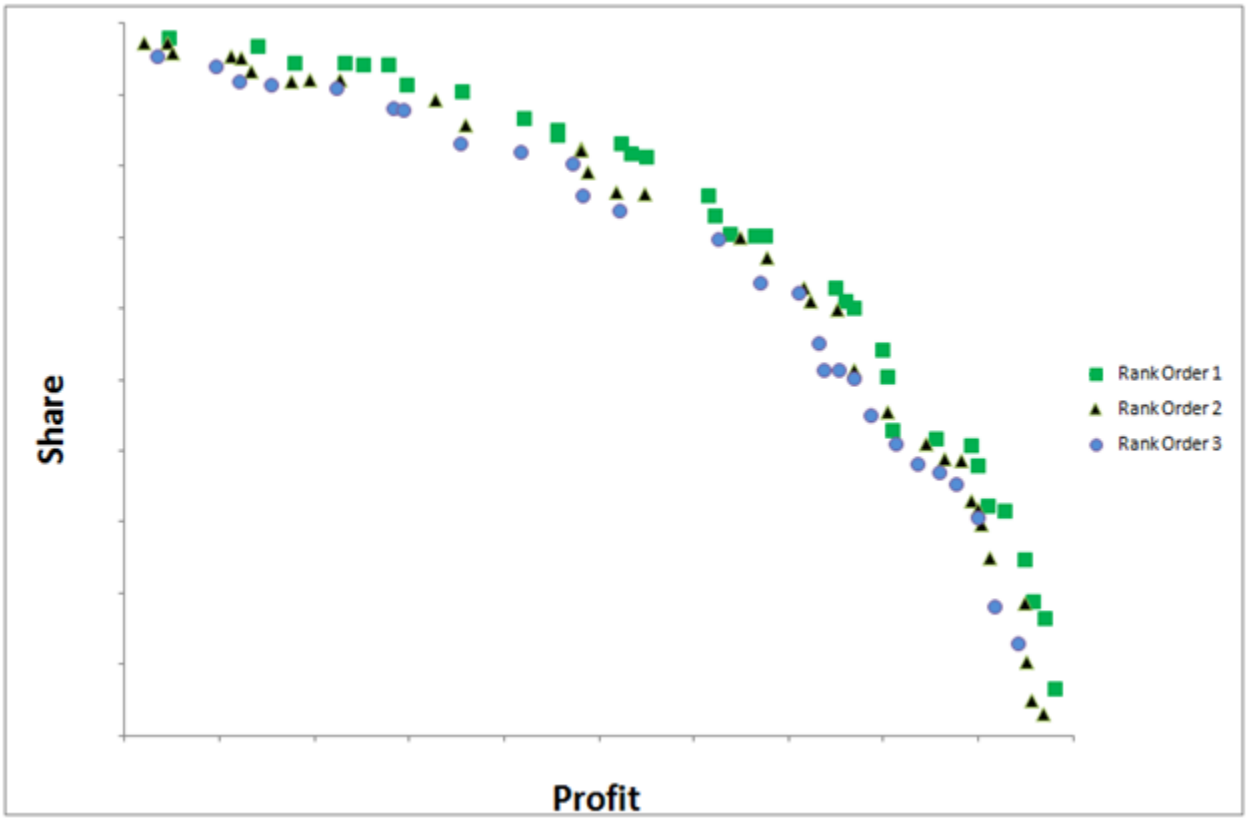
Multi-Objective Genetic Algorithms (MOGAs)

The number of potential products to search often exceeds what exhaustive search can do in a reasonable time. When the search space becomes too big, we can employ Genetic Algorithms (GAs). One of the reasons we employ GAs instead of a more direct hill-climbing routine is that GAs do a much better job investigating a greater variety of near-optimal solutions with respect to the objective criteria that is necessary for a thorough multi-objective search. (For a discussion of GAs, please refer to *http://www.sawtoothsoftware.com/download/techpap/asmtech.pdf*.)

With GAs, pairs of fit product concepts "mate" via the operators of selection, cross-over, and mutation to produce offspring. Both the parent products and the offspring are then compared in terms of fitness. Traditionally, the fitness criterion is just a single dimension, such as share, revenue, or profit. But with multi-objective GAs (MOGAs), we use the rank order criterion to determine the fitness (with additional consideration given to *crowding distance*, described further below). For example, in each generation of the GA, the n products with the most favorable rank order are retained and the remaining are culled. For example, if only the solutions of rank order 3 or better are retained in our example[1], this represents the product solutions shown in Figure 7.

[1] Although we previously described identifying these top 3 rank order solutions based on an exhaustive search of all 750 possible solutions, imagine now that there are hundreds of millions or more potential solutions and that a heuristic GA has been used to generate just a tiny subset of all possible solutions. Among that tiny subset, we've identified the top 3 rank order solutions shown in Figure 7.

Figure 7

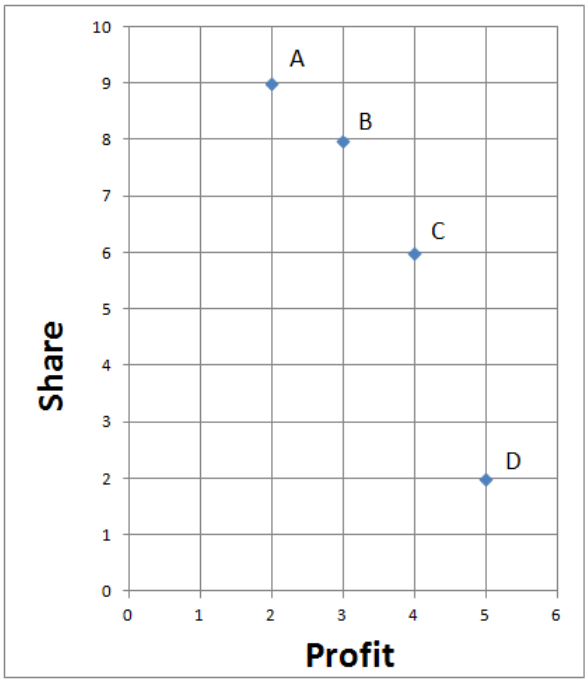


Diversity Preservation and Crowding Distance

As discussed by Ferguson and Foster, "Diversity preservation is significant for two reasons. First, it is desired that the final solution is spread across the performance space. Second, it provides a tie-breaker during selection when two designs have the same non-domination rank." For example, let's imagine that 150 solutions are to be culled from 450 existing solutions to identify the parents retained in the next generation of the genetic algorithm. If only 250 rank order 1 products exist, all are taken forward. That gives us 250 of the 300 solutions to carry to the next generation. Let's imagine that 100 rank order 2 solutions exist, but only 50 of them can be taken forward to the next generation to complete our selection of 300 total most fit products. The 50 rank order 2 solutions with the highest crowding distance (most "elbow room") are selected. A high crowding distance indicates that this solution is more diverse (unique) in terms of the multiple objectives than other solutions. As described by Ferguson and Foster, "…the crowding distance around each design is calculated by determining the average distance of two points on either side of this design along each of the objectives."

Consider four adjacent solutions with the same rank order of domination shown in Figure 8. Before computing the crowding distance, share and profit need to be normalized so that both dimensions have common scaling.

Figure 8



Visually, one can easily see that D is less crowded (more unique in terms of the objectives, with a larger crowding distance) than the other solutions. The crowding distance for each solution is calculated similar to city block distance (also known as *Manhattan distance*) from its two nearest neighbors:

B is one unit of share and one unit of profit different from A, and two units of share and one unit of profit different from C. The average of those four distances is the crowding distance for B: AVERAGE (1, 1, 2, 1) = 1.25.

C is two units of share and one unit of profit different from B, and four units of share and one unit of profit different from D. The average of those four distances is the crowding distance for C: AVERAGE (2, 1, 4, 1) = 2.00.

A and D are endpoints without solutions on adjacent sides. So, we compute the crowding distance with respect to the one nearest point. A is one unit from B both in terms of share and profit, for a crowding distance of AVERAGE (1, 1) = 1.0. D is 4 units from C in terms of share and one unit from C in terms of profit, for a crowding distance of AVERAGE (4, 1) = 2.5.

To summarize, the crowding distances for these four solutions (all belonging to the same rank order of domination) are:

A	1.00
B	1.25
C	2.00
D	2.50

Assuming A, B, C, and D share the same rank order of domination, D>C>B>A in terms of selection priority to survive into the next generation of the genetic algorithm.

Computing Likelihood of Mating

Another aspect of a GA is to determine which of the surviving solutions (that were selected based primarily on rank order of domination and, in the case of ties, secondarily based on crowding distance) "mate" to generate new "offspring" solutions. In single-objective GAs, surviving solutions "mate" with likelihood proportional to their fitness on the single objective, such as profit. Crossover and mutation operations are performed to generate offspring (new solutions). Different strategies could be enacted in MOGAs to select which surviving solutions to pair for the purpose of generating hopefully even more fit offspring. The strategy we use is to select parents among surviving solutions with likelihood proportional to the product of the inverse of the rank order times the crowding distance.

For illustration, just five solutions (among the 300 solutions chosen to survive into the next generation) with rank order of domination and crowding distances are shown in Table 1. The likelihood of selecting each solution to mate is proportional to the values in the final column.

Table 1

Solution	(A) Rank Order of Domination	(B) Inverse of Rank Order of Domination	(C) Crowding Distance	(B * C) Relative Likelihood of Mating
A	2	1/2	0.3	0.15
B	1	1/1	0.1	0.10
C	3	1/3	0.3	0.10
D	1	1/1	0.3	0.30
E	2	1/2	0.4	0.20
. . .	. . .	. . .	. . .	. . .

This approach rewards quality and diversity.

Breakout Criterion

Genetic algorithms can iterate indefinitely with very little or no additional improvement from one generation to the next, so the researcher must specify a breakout criterion. One approach is to specify a fixed number of generations, such as 20 or 50. Another approach that has proven useful in multi-objective search is to measure the increase in hypervolume (the area under the efficient frontier) of the rank-order 1 solutions across, say, the last five generations, and to break out if it ceases to increase by some small threshold (such as 0.5%). The hypervolume calculation is commonly used in evolutionary multi-objective search algorithms. We use the HSO (Hypervolume by Slicing Objectives) algorithm as described by While et al. (2006).

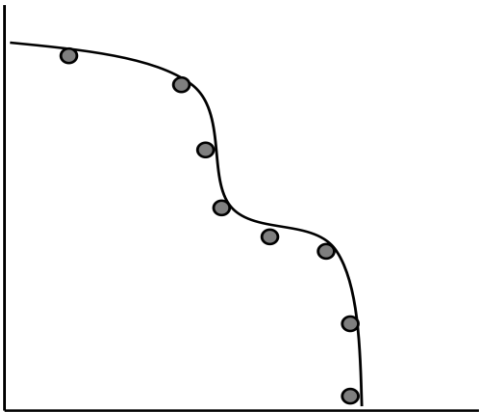
Why Not Just Weight the Multiple Objectives?

As I was listening to Ferguson and Foster's presentation at the 2013 Sawtooth Software Conference, my first inclination was to wonder why one cannot simply create a new objective which is a weighted combination of the multiple objectives (e.g. compute a new variable that represents share x profit). This would mean we simply could continue to use existing algorithms and software (such as ours at the time) whose goal was to maximize a single objective function.

Ferguson and Foster went on to describe multiple concerns with creating a weighted objective function, including:

- The difficulty of finding non-dominated solutions in non-convex regions of the Pareto efficient set. Weighted functions assume a well-behaved smooth convex function, whereas the actual shape of the efficient frontier may be concave over the entire multi-dimensional space or for certain regions within the space (as shown in Figure 9).

Figure 9



- The difficulty of coming up with the right *a priori* weights that meet the organization's goals. Should one equally weight profit and share, or give more weight to profit than share?
- A weighted multi-objective function won't necessarily lead to an uncrowded spread of points across the performance space (they can tend to lack diversity with respect to the goals). According to Ferguson and Foster, "solutions will often clump in the performance space and provide very little information about the possible tradeoffs elsewhere."

More than Two Dimensions

MOGAs can involve more than just two dimensions. We can also determine the crowding distance, rank order of solutions on the dominance criterion, and can compute hypervolume in 3- or 4-space (say, share, profit, revenue, and cost). The same logical steps would apply, but in higher dimensions.

References

Ferguson, Scott and Garrett Foster (2013), “Demonstrating the Need and Value for a Multi-Objective Product Search,” Sawtooth Software Conference Proceedings, pp. 275-304, Orem, Utah.

Orme, Bryan (2010), “Market Simulators for Conjoint Analysis,” downloaded from *http://www.sawtoothsoftware.com/download/techpap/introsim.pdf*

Roy, Proteek Chandan, Kalyanmoy Deb, Md. Monirul Islam (2018), “An Efficient Nondominated Sorting Algorithm for Large Number of Fronts,” IEEE Transactions on Cybernetics.

Sawtooth Software (2003), “Advanced Simulations Module (ASM) for Product Optimization v1.5 Technical Paper,” downloaded from *http://www.sawtoothsoftware.com/download/techpap/asmtech.pdf*

While, Lyndon, Phil Hingston, Luigi Barone, and Simon Huband (2006), "A Faster Algorithm for Calculating Hypervolume," IEEE Transactions on Evolutionary Computation, Vol 10, No 1, February. Accessed 2/21/2018 at http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=3022&context=ecuworks.