Product Portfolio Evaluation Using Choice Modeling and Genetic Algorithms

Chris Chapman, PhD
James Alford, PhD

Microsoft Hardware

- PC accessories sold worldwide through retail and PC makers
- Product design and management in Redmond, Washington

- *Specific product line and attributes are disguised here*

The problem space

- Given conjoint analysis data …

1	773	28	-0.237	-0.351	0.588	-0.312	-0.397	0.431	0.278	0.981
2	797	28	-0.513	-0.104	0.618	2.057	-0.966	-0.146	-0.944	3.685
3	724	28	-0.852	0.666	0.185	-2.546	0.186	1.033	1.327	0.088
4	803	28	-0.396	0.435	-0.039	5.356	-1.503	-1.644	-2.209	0.743
5	532	28	-0.334	0.337	-0.003	-3.71	1.422	1.33	0.958	-0.336
6	728	28	-0.786	0.469	0.317	0.518	-0.399	0.151	-0.27	0.42

- We know how to optimize a product

- But what about a product *line*?

- If we knew about potential ideal lines, what could we do?

Business questions

- We make X# products in a category …
  How many products *should* we make in the category?
- Some people buy feature Y and some don't …
  How many *can we expect to* want feature Y in an optimal portfolio?
- We make products with such-and-such feature sets …
  Are there feature sets (products) we are *missing*?
- Current retail price points are A, B, C …
  Do those price points match the optimal products?

Intuition

- Suppose we can derive a putative optimal line from data …

- Sampling is not perfect
  Respondents do not answer perfectly
  Estimation will not fit the data perfectly
  Choices do not perfectly predict behavior

- **Implication**:
  A single result will be imperfect

- Use near-optimal line as a hypothesis to *explore further*
- *Repeat multiple times* to get a sense of generalizability

Method

Overview of the approach

- Collect CBC or ACBC data for a product category
- Derive individual-level part worths using HB model

- Iterate to fit *many* portfolio preference models:
  - Sample some of the data
  - Find a near-optimal portfolio to fit ← **How?**
  - Assess performance on the holdout data
  - Performance = Total Preference share vs. competition and "none"

- Across the many models, inspect:
  - **Size**: how does preference increase with #products?
  - **Features**: how many people want each feature?
  - **Products**: are there gaps vs. current portfolio?

Finding a near-optimal portfolio

- Given several attributes with several levels …
  Many possible products, which combine for
  Exponentially many portfolios

- For our problem:
  9 attributes with 2-7 levels ➔ 1080 possible products

- For *K* products: $NofPortfolios = \frac{(NofProducts)!}{(NofProducts-K)!K!}$

- With 1080 products and *K*=10, *NofPortfolios* ≈ 10^{23}

- **Implication**:
  Use a method that can search a large space ➔ **Genetic Algorithm**

Genetic Algorithms

Genetic algorithm overview

Preliminary

Represent solution in terms of discrete parts, aka “genes”

From features to a list of candidate portfolios

- ***Product*** = list of attribute/feature pairs Attr1Feat2 + Attr2Feat1 + Attr3Feat2

From features to a list of candidate portfolios

- ***Product*** = list of attribute/feature pairs
- Each attribute/feature maps to part worths located in a specific ***column***
- *Product =* vector of the column positions that represent its features

Attr1Feat2 + Attr2Feat1 + Attr3Feat2

Col 2 + *Col 5 +* *Col 10*

[2,5,10]

From features to a list of candidate portfolios

- ***Product*** = list of attribute/feature pairs
- Each attribute/feature maps to part worths located in a specific ***column***
- *Product =* vector of the column positions that represent its features
- ***Portfolio*** = a set of products

Attr1Feat2 + Attr2Feat1 + Attr3Feat2

Col 2 + *Col 5 +* *Col 10*

[2,5,10]



[2,5,10] + [1,5,9] + [2,6,10] + …

From features to a list of candidate portfolios

- ***Product*** = list of attribute/feature pairs

Attr1Feat2 + Attr2Feat1 + Attr3Feat2

- Each attribute/feature maps to part worths located in a specific ***column***

Col 2 + Col 5 + Col 10

- *Product* = vector of the column positions that represent its features

[2,5,10]

- ***Portfolio*** = a set of products

[2,5,10] + [1,5,9] + [2,6,10] + …

- ***Candidates*** = a stack of portfolios, each with several products

#1: [2,5,10] + [1,5,9] + [2,6,10] + …
#2: [1,5,9] + [2,6,10] + [1,6,9] + …
….

Genetic algorithm overview

Preliminary

Represent solution in terms of discrete parts, aka "genes"

```
Feature columns
List of products


Prod 1   =  1  4  9  11  15  19  …
Prod 2   =  2  5  8  11  14  22  …
…
```

Genetic algorithm overview

Start

Create random set of candidate portfolios

```
1 4 9 11 15 19        Feature columns
2 5 8 11 14 22 …      List of products
```

Preliminary

Represent solution in terms of discrete parts, aka “genes”

Genetic algorithm overview

Start

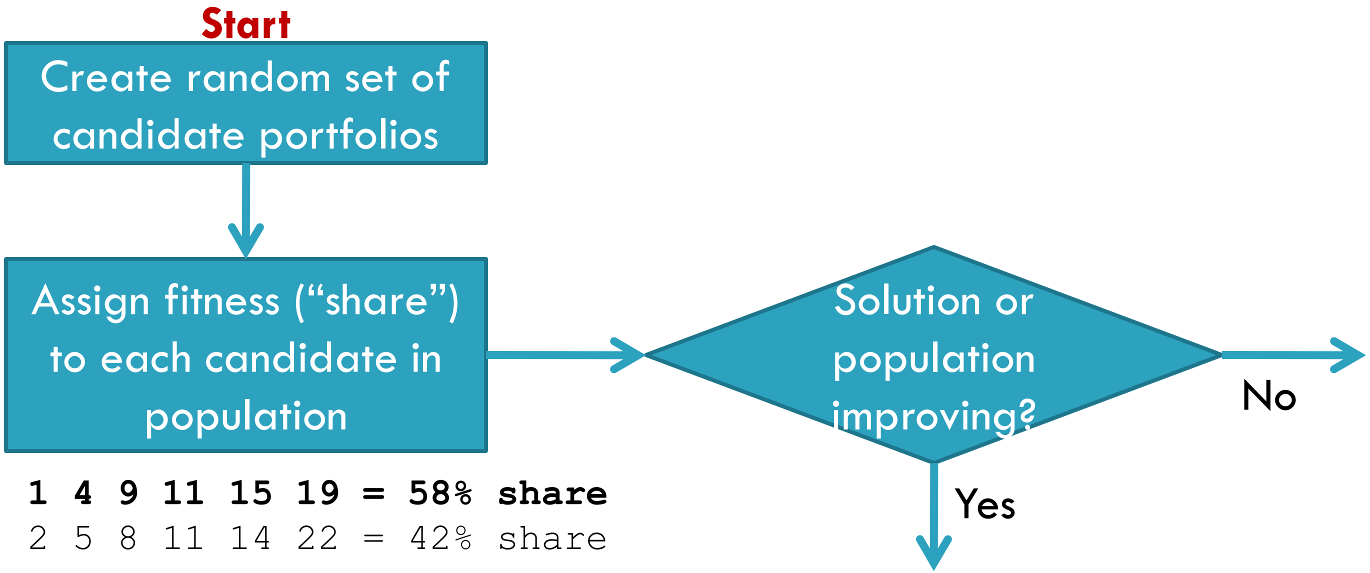
Create random set of candidate portfolios

```
1 4 9 11 15 19
2 5 8 11 14 22 …
```

Assign fitness ("share") to each candidate in population

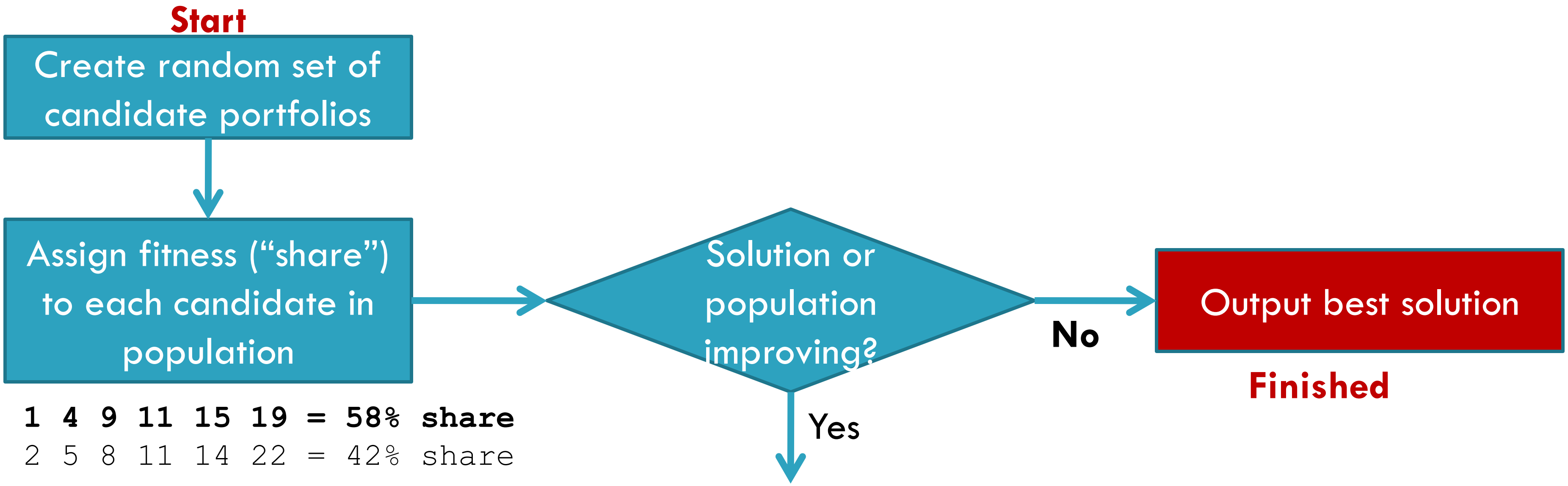
```
1 4 9 11 15 19 = 58% share vs. fixed or "none"
2 5 8 11 14 22 = 42% share vs. fixed or "none"
```

Genetic algorithm overview



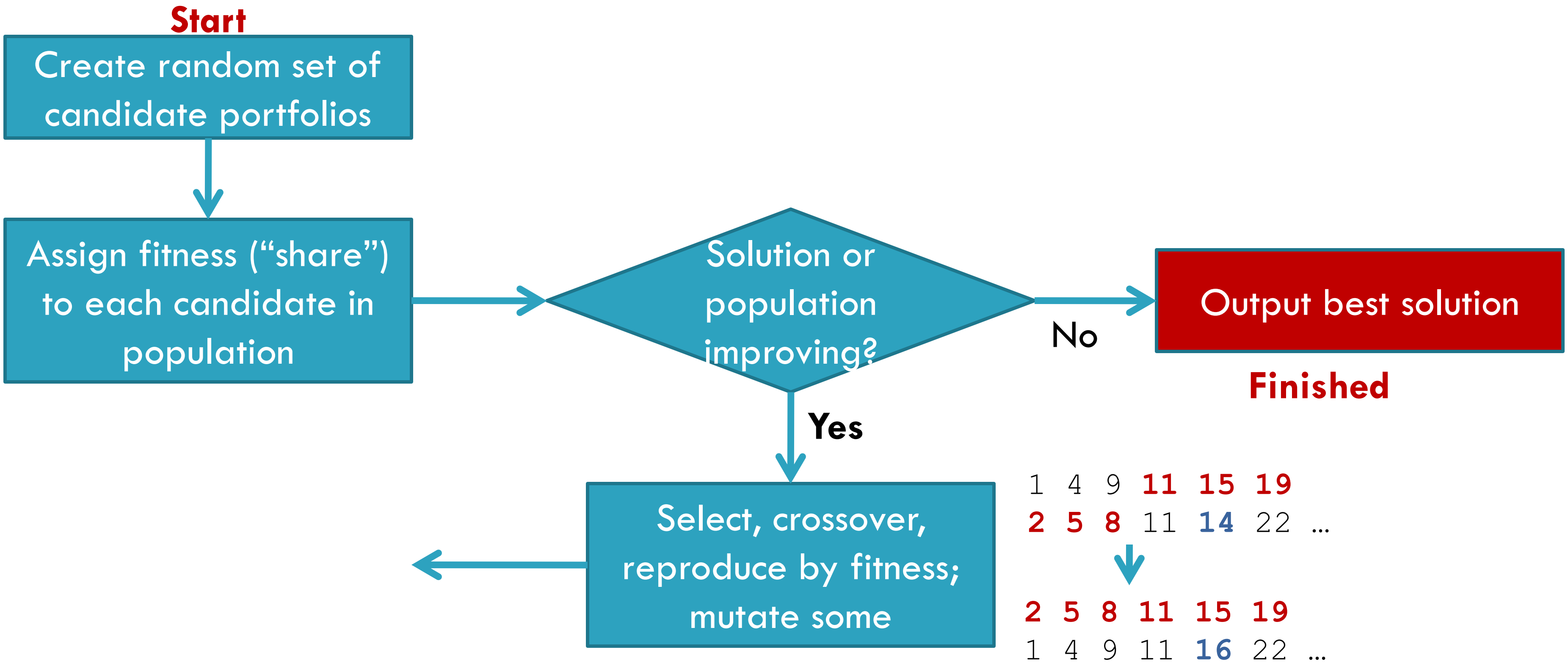

```
1 4 9 11 15 19 = 58% share
2 5 8 11 14 22 = 42% share
```

Genetic algorithm overview

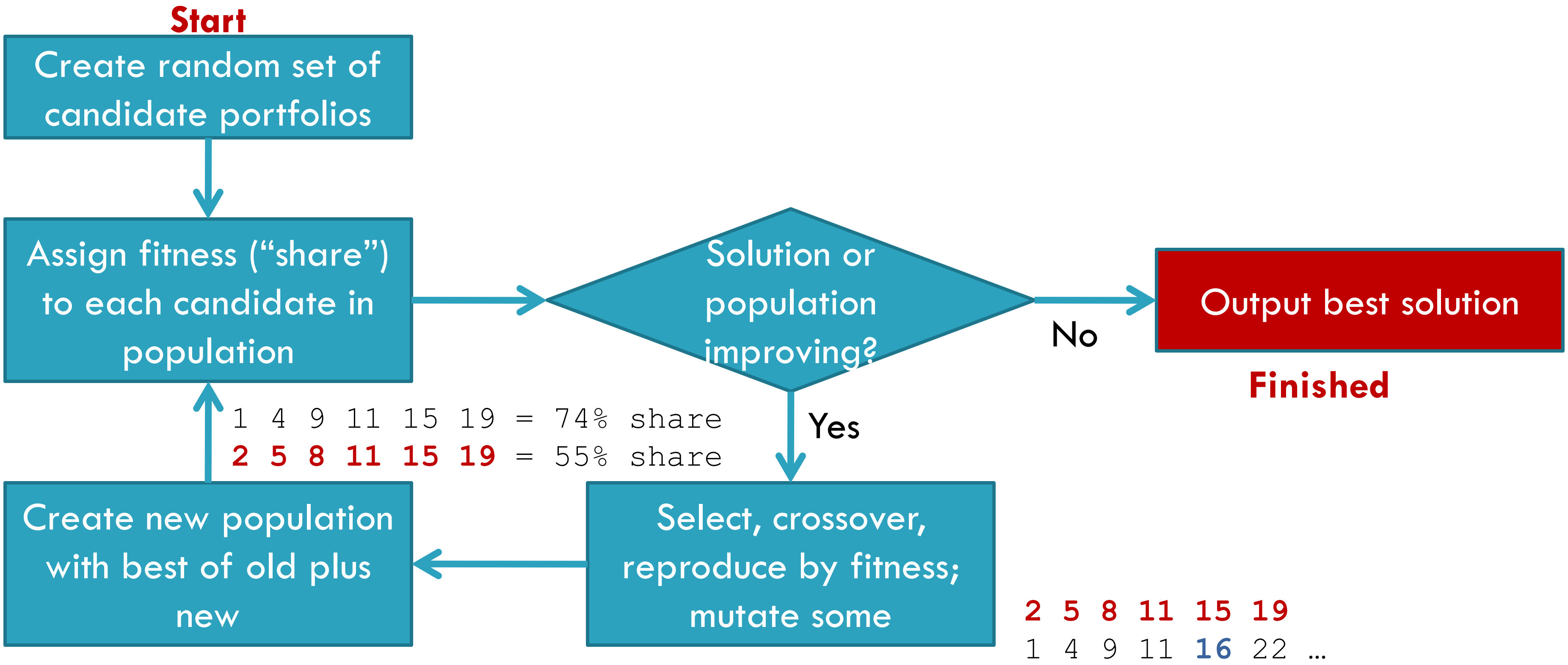



```
1 4 9 11 15 19 = 58% share
2 5 8 11 14 22 = 42% share
```

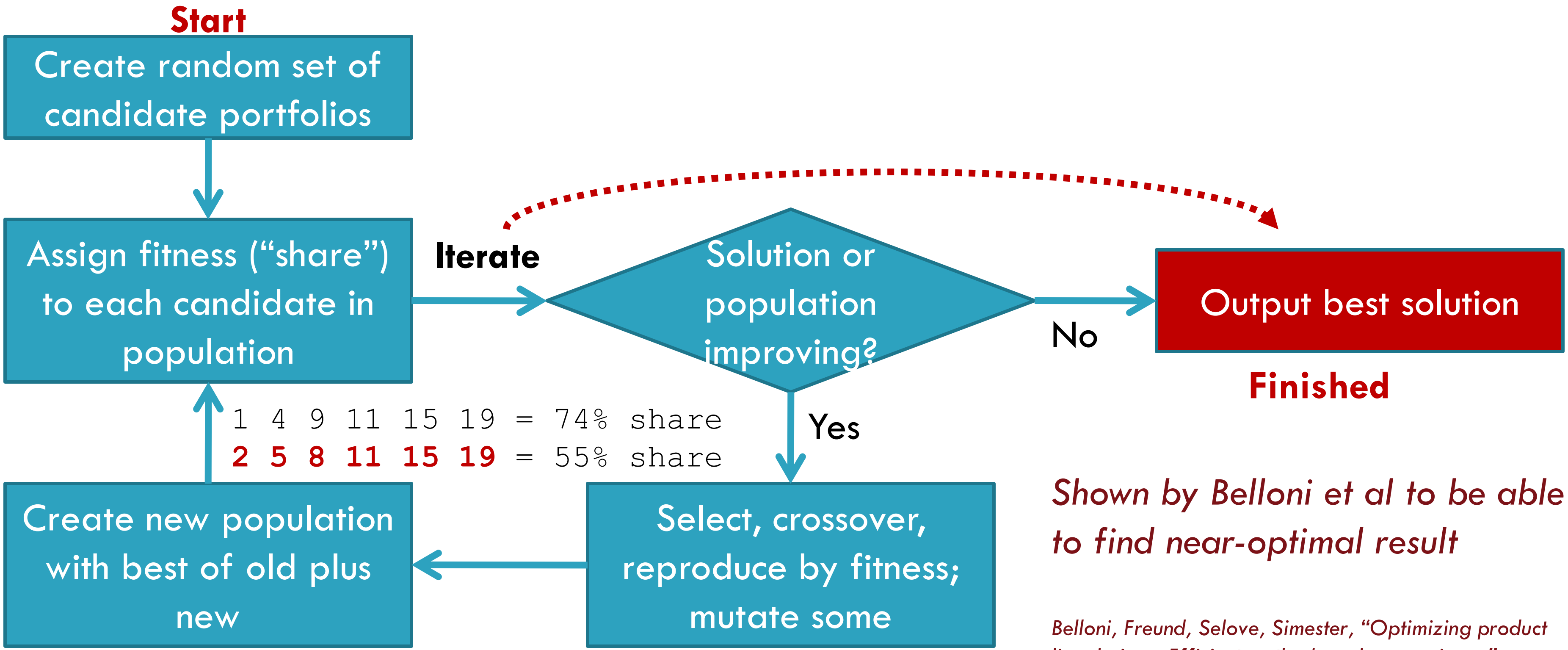
Genetic algorithm overview



Genetic algorithm overview



Genetic algorithm overview



Shown by Belloni et al to be able to find near-optimal result

Belloni, Freund, Selove, Simester, "Optimizing product line designs: Efficient methods and comparisons," Management science 54, no. 9 (2008).

Details

- Genome definition:

  Allele x_n in [col_{start}, col_{end}] = 1 product attribute

  Gene = collection of alleles = 1 product in portfolio = [x_1, x_2, … x_q]

  Genome = [$gene_1$, $gene_2$ … $gene_k$] = portfolio of products (k = portfolio size)

- Data
  - Per-respondent part worth estimates from Sawtooth Software CBC and ACBC studies with hierarchical Bayes estimation
  - **N=716 CBC & N=405 ACBC**, US online samples
  - Bootstrap sampled 60% for model development, 40% holdout on each GA run
  - Total 9 attributes with 2-7 feature levels each

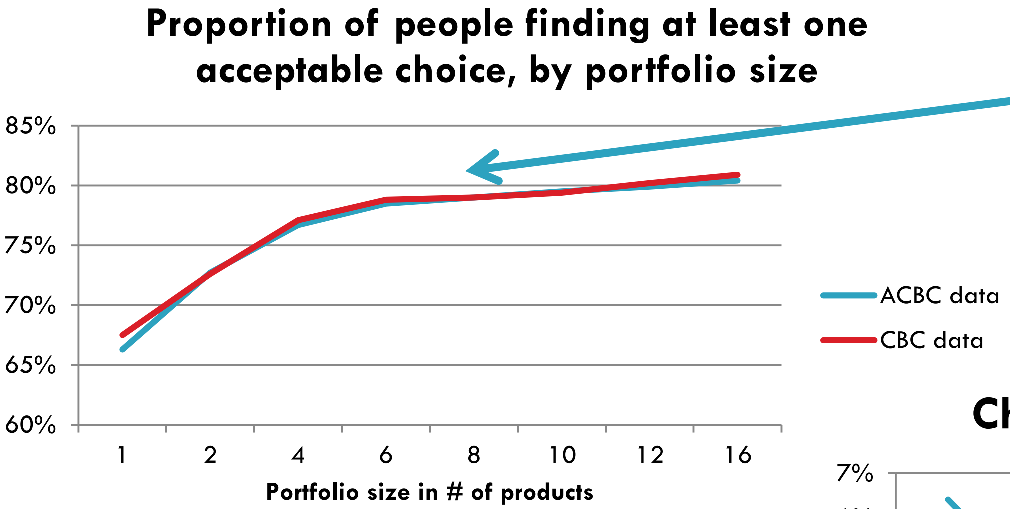
- Algorithm & parameters
  - RGenoud algorithm from UC Berkeley, version 5.4-7
  - Solution represented as vector of integers mapped to columns, i.e., length of (8 integers/product) × (portfolio size)
  - GA population size = 400, Maximum generations = 50, Wait generations = 10
  - Operators = equally divided among: Cloning, Uniform Mutation, Boundary mutation, Non-Uniform Mutation, Simple Crossover, Whole Non-Uniform Mutation, Heuristic Crossover

- Fitness
  - Fitness function = **total product share vs. "none" for portfolio**, in development sample
  - Based on conjoint analysis data (hierarchical Bayes logit model, main effects only, per respondent)
  - Reported results = fitness performance of GA solution in holdout sample

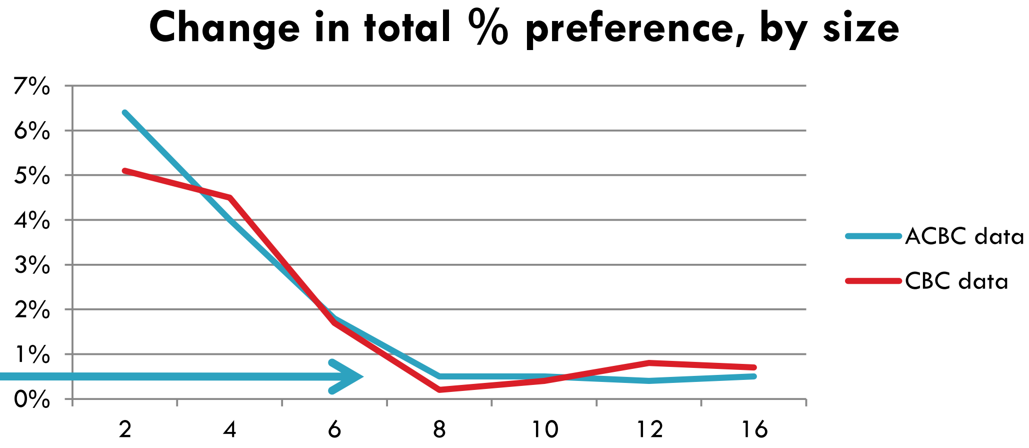
- Repetitions
  - **50 GA runs each** with new sampling for (**CBC + ACBC** datasets) × (**k=1,2,4,6,8,10,12,16,20** products per portfolio)
    50 runs × 2 data sets × 9 sizes = 900 total "best portfolios" selected from space of ≈18,000,000 portfolios searched

Findings

Q: What portfolio size meets users' needs?



Sharply diminishing return in total preference for k>6 products

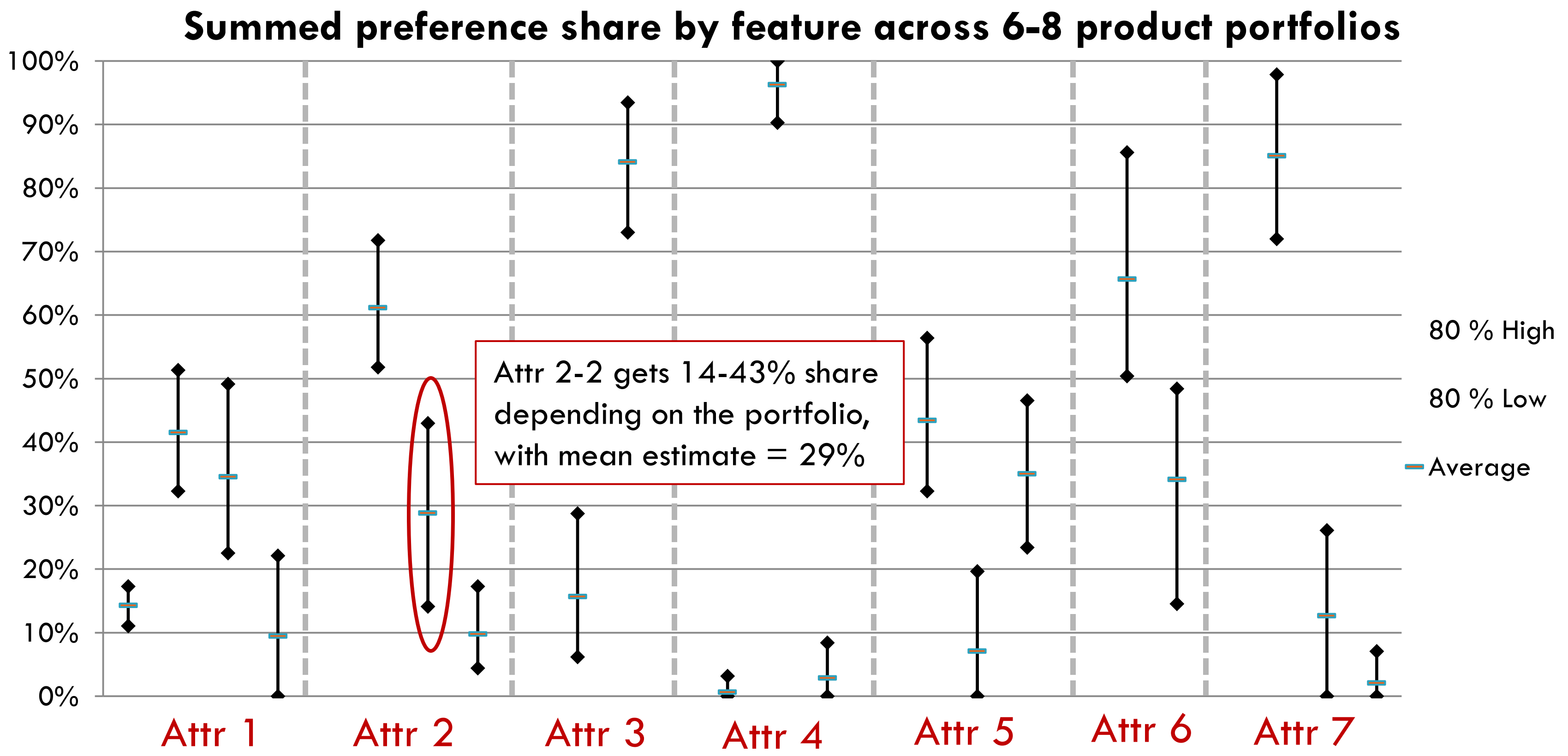


Additional products above k>6 yield less than 1% additional *preference* share per product

Q: What is the range of preference by feature?

- Suppose we have an attribute of particular interest:
  E.g., Attribute 2/Feature level 2
- MNL estimates preference, but does not account for limits of portfolio optimization
- Estimate *Feature demand* | *Portfolio structure* within preferred portfolios
- Demand(feature|portfolio) = $\sum_{i=1}^{k} \begin{pmatrix} \text{if Feature in prodi: } Pi\ preference\ share \\ \text{otherwise: } 0 \end{pmatrix}$
- **Example**:
  Attr2/Feat 2 has 35% MNL share, but it might differ in an optimal portfolio.
  What would it be in a near-ideal portfolio?

Q: What is the range of preference by feature?



(brand and price not shown)

Q: Are there specific product opportunities?

List the products by frequency across portfolios

Are there products that often appear, but we don't make?

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

Attr 2 Attr 6

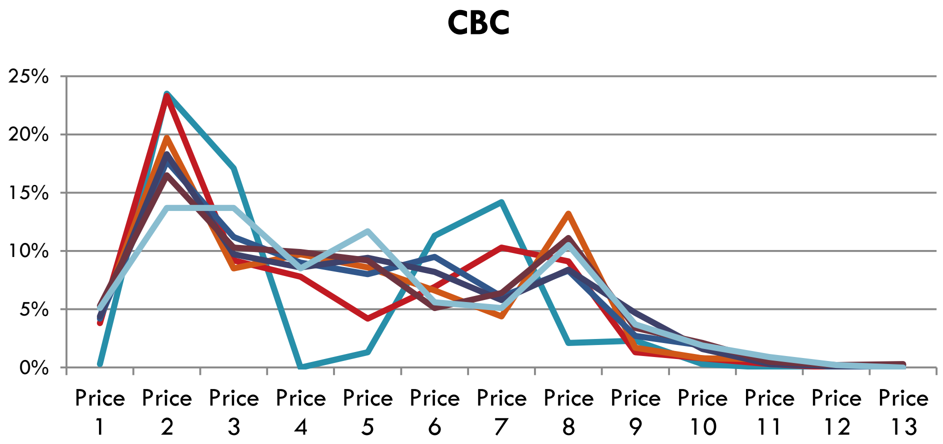
Two products often appear that are not part of our portfolio

The key is the combination of attributes 2 + 6

Q: Are there commonly-appearing price bands?

- Less interest than we had expected at Price 1 and Prices 11-13
  ➔ customers less interested in minimal or maximal products,
  but want a mix of features at well-defined price points

- Revised our concept of “good / better / best” lineup in this category



Conclusions

- Don’t make more than 6-8 products in this category (unless the cost is less than the value of 1% share)
- Knowing how many people *should* be interested in each feature ➔ *target underperforming features*
- Investigate product gaps that appear in optimal portfolios
- Ensure price point concepts match the portfolios’ demand
- *Do more of this kind of modeling! (It works with existing data)*

Discussion

Questions and limitations

- Are the results stable across datasets and categories? Can we reliably aggregate portfolios in this way?
- How do IIA issues play into the aggregation?
- How does this approach relate to others, e.g., from financial portfolio models?
- **Recommendation:**
  Use for hypothesis generation, not for "the answer"
- Computationally very intensive:
  can take days to run on a multicore machine

Availability of the code

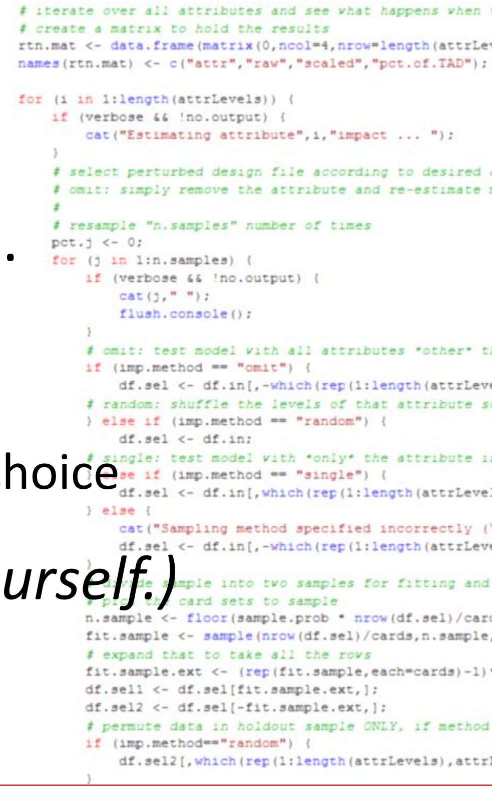
- Complete code example available:
  chris.chapman@microsoft.com

- Written in **R**. *Must be customized* for your problem.

- Options:
  - Use HB draws; Gumbel error; bootstrapping; tuning
  - Preference by logit share, first-choice, roulette-draw first choice

(Note: research code has no warranty; evaluate for yourself.)

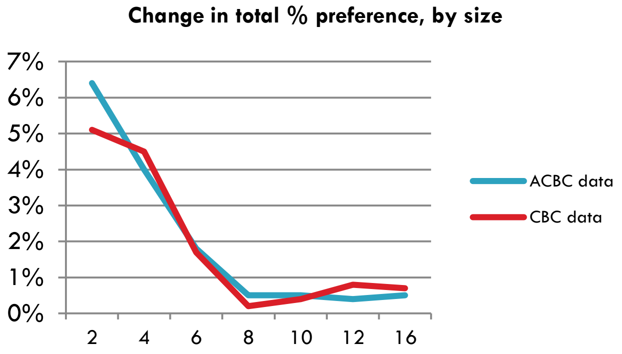
- **Thank you!**

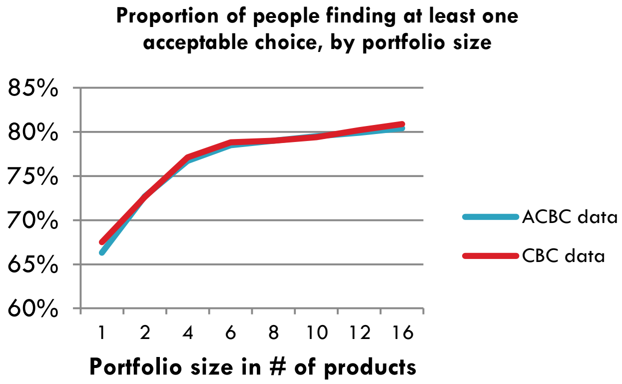


Appendix: CBC vs. ACBC Observations

CBC vs. ACBC

- Strikingly similar results on portfolio size
- ACBC used smaller sample (but did not try the reverse with CBC sample size)



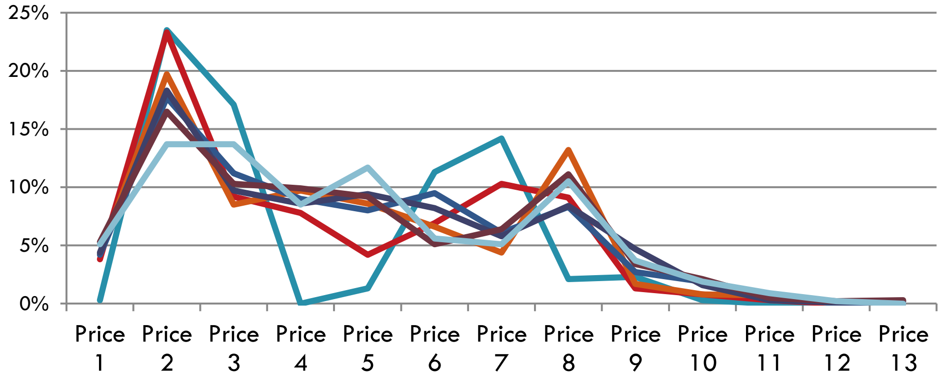


CBC vs. ACBC

- Strikingly similar results on portfolio size
- ACBC used smaller sample (but did not try the reverse with CBC sample size)
- ACBC had more consistency than CBC on price banding

Distribution by price & portfolio size (ACBC)

CBC



CBC vs. ACBC

- Strikingly similar results on portfolio size
- ACBC used smaller sample (but did not try the reverse with CBC sample size)
- ACBC had more consistency than CBC on price banding
- Conclusion:
  ACBC data appears to be at least as good as CBC for this
  ACBC may have a slight edge
  - Stakeholder face validity
  - Smaller samples needed
  - Respondent engagement
  - Cleaner results across price banding in this study