

AUTOMAGICAL: GENETIC ALGORITHMS IN FINANCE

Jacob Loveless

Overview

- The importance of framing the problem
 - ▣ Maximum Sum Subarray example
 - ▣ Genetic Algorithms and Memetic Algorithms
- Using GA's to build multiple specific models
- Using GA's to build a general model

Problem Framing

- Often, we find that relatively simple problems in one dimension become extremely difficult in higher dimensional spaces. Richard Bellman called this the “curse of dimensionality” which describes the problem caused by the exponential increase in volume associated with adding extra dimensions to a (mathematical) space.
- For example, given one-dimensional array (\mathcal{A}) of numbers- imagine we are given the computational task of finding the contiguous subarray with maximum sum. As in

array \mathcal{A} consists of the sequence of values $-2, 1, -3, 4, -1, 2, 1, -5, 4$.

Then the contiguous subarray with the largest sum is

$-2, 1, -3, \underline{4, -1, 2, 1}, -5, 4$

which sums to 6.

- In 1984, Jay Kadane of Carnegie-Mellon developed a linear time solution to this one dimensional problem using a simple example of dynamic programming.
- But, assume we increase the dimensionality, and ask the following: given a table M , compute the maximum sum of column \mathbf{F} using boundaries of A,B or A&B. For example

Problem Framing

given a table M , compute the maximum sum of column **F** using boundaries of A,B or A&B

Table M

A	B	F
1	2	-2
2	2	1
1	3	-3
3	1	4
5	2	-1
3	3	2
3	1	1
1	2	-5
4	3	4

Problem Framing

Sample Boundary Condition (not optimal):

$$A \geq 2, A \leq 4, B \leq 2$$

Which Yields

A	B	F	$A \geq 2$
1	2	-2	
2	2	1	X
1	3	-3	
3	1	4	X
5	2	-1	X
3	3	2	X
3	1	1	X
1	2	-5	
4	3	4	X

A	B	F	$A \leq 4$
1	2	-2	X
2	2	1	X
1	3	-3	X
3	1	4	X
5	2	-1	
3	3	2	X
3	1	1	X
1	2	-5	X
4	3	4	X

A	B	F	$B \leq 2$
1	2	-2	X
2	2	1	X
1	3	-3	
3	1	4	X
5	2	-1	X
3	3	2	
3	1	1	X
1	2	-5	X
4	3	4	

And the result is the sum of column F where there is an intersection of the boundary conditions: this occurs at $F[1;3;6]$ which is the sum of (1;4;1) or 6.

A	B	F	$A \geq 2$	$A \leq 4$	$B \leq 2$	Intersection
1	2	-2		X	X	
2	2	1	X	X	X	X
1	3	-3		X		
3	1	4	X	X	X	X
5	2	-1	X		X	
3	3	2	X	X		
3	1	1	X	X	X	X
1	2	-5		X	X	
4	3	4	X	X		

Problem Framing

- What makes this problem that much more difficult? Obviously the addition of the columns raises the dimensionality, but the number of distinct values for those columns raises it further.
- Traditionally, we would say we have 2 dimensions (A and B) with 5 and 3 breakpoints respectively. In this example, were we to brute for the solution we would have 111 possible solutions.
- If we were to add another column, which had 5 breakpoints, the solutions could now include attributes A, B, A&B, B&C, A&C and A&B&C and each value set. The resulting solution space increases to 1791 possible solutions. Below is a table of attributes (column) and values (assuming each attribute has the same number of values) and the size of the search space.

Attributes (Dimensions)	Values (Breakpoints)				
	2	4	6	8	10
1	3	10	21	36	55
2	9	100	441	1,296	3,025
3	27	1,000	9,261	46,656	166,375
4	81	10,000	194,481	1,679,616	9,150,625
5	243	100,000	4,084,101	60,466,176	503,284,375
6	729	1,000,000	85,766,121	2,176,782,336	27,680,640,625
7	2,187	10,000,000	1,801,088,541	78,364,164,096	1,522,435,234,375
8	6,561	100,000,000	37,822,859,361	2,821,109,907,456	83,733,937,890,625
9	19,683	1,000,000,000	794,280,046,581	101,559,956,668,416	4,605,366,583,984,370
10	59,049	10,000,000,000	16,679,880,978,201	3,656,158,440,062,980	253,295,162,119,141,000

Problem Framing

- A 4 value 9 dimensional problem (9 columns, each with 4 values) requires a search space of 1 billion.
- On our system, using modern processors (multiple cores) we can perform these 1 billion queries in about 39 seconds- or about 1 query per .04ms. The problem lends itself well to parallelization (each query can run independent of another) - and the number of rows is not important (the problem's running time is relegated to the search space- not the number of rows).
- However, even with these benefits, a 10 attribute, 10 value brute force calculation would require ~313.24 years to compute.
- The search space can be compressed by “bucketing” values into ranges. For example if the values are evenly distributed from 1-100, placing them in equal buckets in increments of 10 helps. Of course, it helps to know the distribution of the values to build the correct “buckets”. Statistics and heuristics coexist, even if it's an unhappy marriage.

Problem Framing

- Before we delve into the algorithm itself- it's important to ask: **How well does this hypothetical problem relate to practical problems in trading signal development?**
- Almost directly. Consider a time series table T , in which each row represents a new market data update (called a tick). We run a back testing application which populates the column **Return** with the profit or loss resulting from taking a position at that exact time- given a set of parameters (profit objective, stop limit and maximum holding time). For example

time	Return
00:06.4	78
00:06.4	78
00:06.4	156
00:07.3	-312
00:07.4	-156
00:08.2	78
00:08.3	78
00:08.8	78
00:10.3	-156
00:13.0	-156
00:13.1	-312
00:13.6	-468
00:13.6	-468
00:13.7	312

Problem Framing

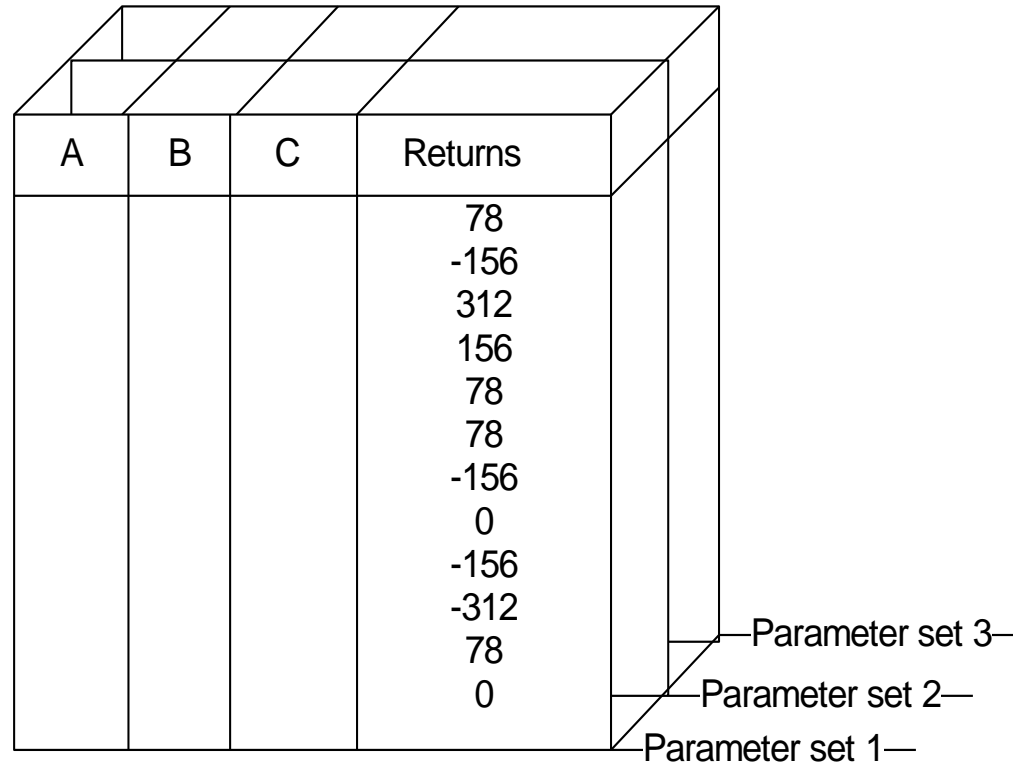
- We then add some attributes which explain the market at that moment, for example

time	market price	Oil Price	S&P Price	USD/GBP	Return
00:06.4	14	108	1403.4	0.58	78
00:06.4	13	116	1400.81	0.52	78
00:06.4	13	117	1408.86	0.51	156
00:07.3	13	106	1407.69	0.53	-312
00:07.4	15	111	1409.09	0.54	-156
00:08.2	12	114	1406.65	0.57	78
00:08.3	14	109	1402.31	0.55	78
00:08.8	13	107	1406.2	0.5	78
00:10.3	12	120	1409.18	0.57	-156
00:13.0	13	118	1408.52	0.6	-156
00:13.1	11	101	1408.84	0.52	-312
00:13.6	11	113	1403.47	0.54	-468
00:13.6	11	107	1410.93	0.59	-468
00:13.7	10	116	1409.81	0.53	312

- And we want to maximize the sum of the returns using some boundary conditions on other variables.

Problem Framing

- Of course, we will make this more complicated: we have many more attributes (1,000+) and values in the 10-50 range.
- In practice, our search spaces are so large, they yield a number whose value has no practical relevance.
- Compounding this initial space, is the concept of parameters (profit objective, stop limit and time limit). The parameters directly affect the **Returns** column- meaning a trade in which the user risks 1\$, attempts to make 1\$ and holds the position for no more than 10 minutes has an entirely different **Returns** vector than the same trade which risks 2\$, attempts to profit 3\$ and holds for 30 minutes. In reality we work on "cubes", not tables- where each axis is a combination of profit objective, stop limit and time limit.



A	B	C	Returns
			78
			-156
			312
			156
			78
			78
			-156
			0
			-156
			-312
			78
			0

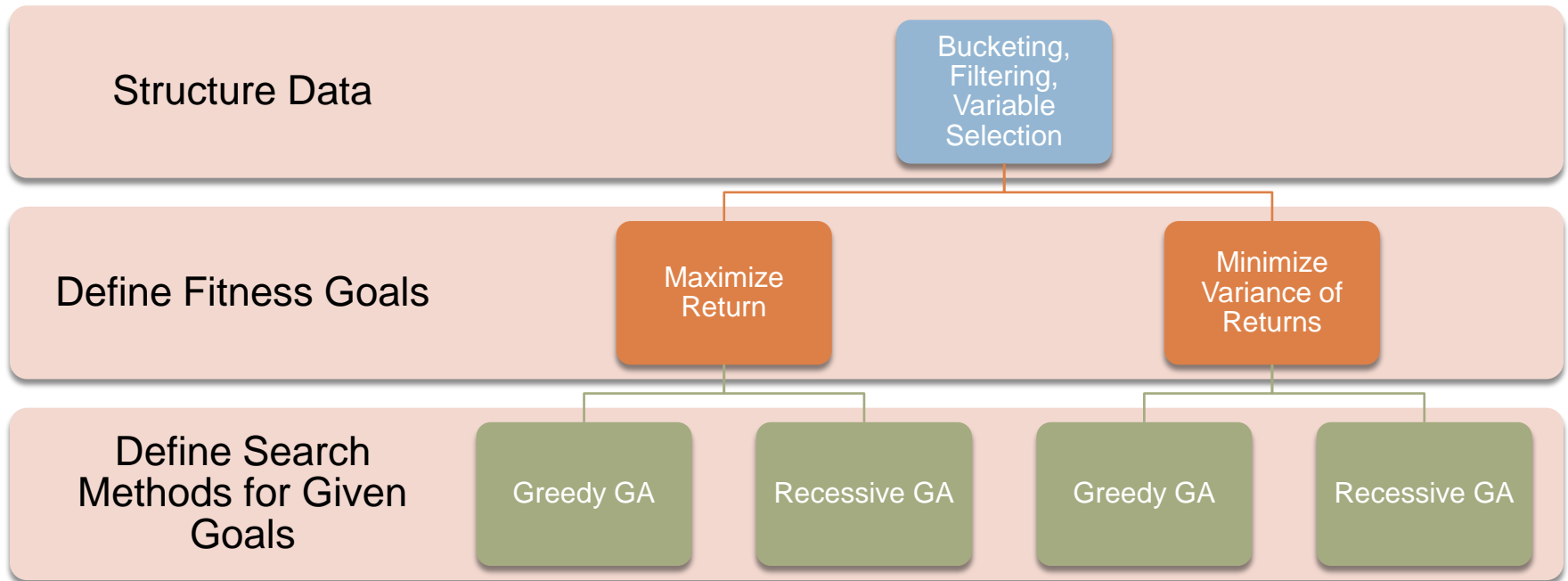
Parameter set 3—
Parameter set 2—
Parameter set 1—

Genetic and Memetic Algorithms

- A memetic algorithm uses a population of agents to seek optimal (or at least very good) solutions to a problem, using a given fitness function which ranks the goodness of the solutions. The agents seek candidate solutions throughout the fitness landscape (search space), using knowledge about the problem to improve the solutions, and cooperating and competing among themselves. Cooperation means that cooperating agents give rise to new agents which share characteristics from them, while competition is achieved by selection pressure over the population of agents. Although this description sounds very much in the manner of conventional genetic algorithms, memetic algorithms take a qualitatively different approach
- We run a memetic algorithm: which is to say we have multiple search methods (traditional Genetic Algorithm, Tabu style, Greedy, SA ...) and multiple fitness goals. Each search method attempts to build a solution (or collection of solutions), and at the end of each "generation" or iteration, it updates a master table of all solutions and their fitness thus far.
- Although the algorithms utilize their own method, at the start of each generation they integrate the results from their peers. For example, an elite style selection algorithm (which combines the best rules from the base) would be able to use those rules generated by a random selection search method. Further to the point, some search systems use different fitness methods entirely. For example, a hybrid genetic algorithm/greedy algorithm (greed1) might score a trading signal based upon the total return (high return). A second instance of that algorithm (greed2) might score a trading signal based upon the variance of the returns (lower risk). At the end of each generation, every fitness score is computed- so greed1 will (and often does) select rules generated by greed2- even though they are optimizing towards different goals. Then end goal being to develop high return, low risk signals.

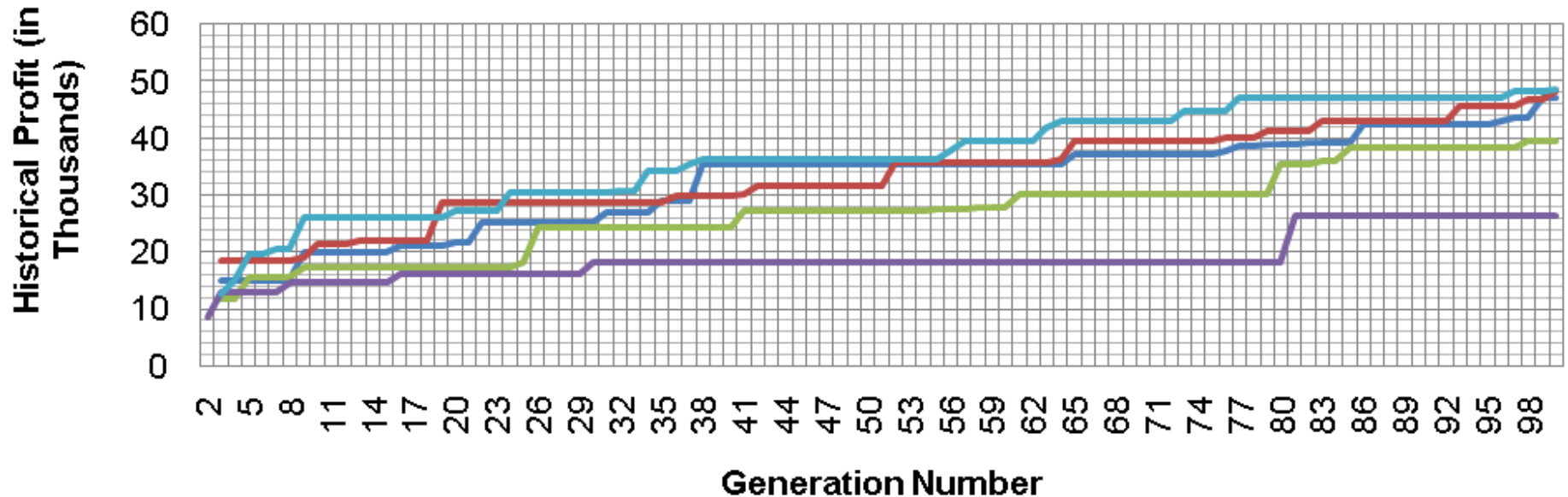
Genetic and Memetic Algorithms

- The basic search “framework” is highly adaptive.



Genetic and Memetic Algorithms

- The next trick is to efficiently allocate resources across the search methods.
- This “meta heuristic” allows for more successful search methods to get larger pools of resources (population sizes).
- This also helps solve some of the issues with GA's, namely plateaus.
- Lastly, this allows one to add search methods with minimal impact (unsuccessful methods are eventually weeded out).
- Convergence is often faster intra method, but some methods directly compete and allow for better mixing.



Using GA's to build multiple specific solutions

- This is the **Fisherman's Net** version of data mining
- We build a large collection of specific rules, and match online data to those rules.
- Often the result is a black box, meaning the rule base is so complicated we can't fully describe it
- This works well in certain problems:
 - ▣ Micro market trading signal development
 - ▣ Market data filtering (detecting illquidity)
- It also helps in factor analysis. One can look at the attributes used by the population and determine those attributes which are the most descriptive

Using GA's to build general solutions

- This is the **Fisherman's Pole** version of data mining
- We use a GA to fit the data to a predefined formalism
- This is helpful in “big picture” models
 - Risk Management and Betting Systems
 - “Mark to Model” situations
 - Longer term estimations of value
- An example is using the framework as an ODE solver., for example an S-System
 - An enormous variety of nonlinear differential equations and functions have been recast exactly in the canonical form called an S-system, which was developed for systems modeling in computational biology (see. M. Savageau and E. Voit). An S-system follows the form:

$$\dot{X}_i = \text{Alpha}_i \prod_{j=1}^n X_j^{g_{ij}} - \text{Beta}_i \prod_{j=1}^n X_j^{h_{ij}} \quad i = 1, 2, 3 \dots n$$

- Which becomes a problem of parameter optimization. The end result is a complex dynamical system which describes a given set of attributes (e.g. a portfolio of assets, a portfolio of signals, a collection of attributes)

Questions

