LECTURE 5: Rugged Landscape models and Simulated Annealing, Genetic Algorithm Search models

Rugged fitness landscape models are found everywhere in science and social science.

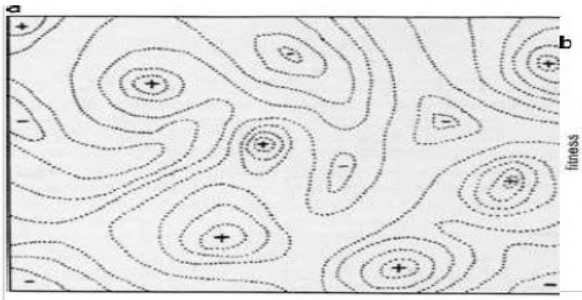
Business organization – adapting an organization to a changed economic environment. Lots of choices of who to buy from in supply chain; how much RD to do and on what; whether to be niche quality producer or lowest priced in market etc. Lots of choices, with landscape defined in terms of profits. Strategic management courses.

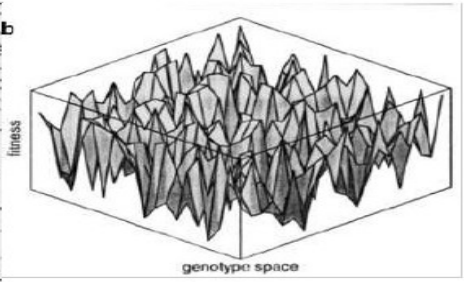
Hiring, admission, and dating algorithms – getting the best match, Say there are 10 attributes that matter for doing a good job. You categorize them as 0-1 --- job candidate has attribute or doesn't. If CONFIGURATIONS -- the mix of attributes -- matter, there are a lot of cases to explore. Configurations enter because someone with say energy and basic knowledge but little experience may dominate someone with lots of experience and knowledge but limited energy. Configuration of 10 attributes represented on row vector 00011 10101 has fitness, say 0.5; 10010 01100 has another fitness 0.3. This gives 2^{10} configurations to explore – 1024 so computer help is much appreciated – to find which configuration gets you best employee**.** Landscape fitness defined in performance on job and turnover.

Engineering example … Modeling Spacecraft Design Activities as Rugged Fitness Landscapes Chiesa, Raytheon, Nov 2018) Mars Rover Design Space Model . Fitness defined as number of samples collected per mission versus mission cost. Select 7 variables to be included in evaluation of Rover Design model. For each variable N there were 2 different values. So there will be $2^7 = 128$ solutions. Simulation to compare the 128 possible choices.

Biology evolution most important. Sewell Wright introduced picture as device for thinking about evolution. Huge literature – lots of theory, limited evidence bcs hard to assess landscape from data/experiments. Value in assessing disease – bacteria vs viruses – finding meds for flu virus that mutate fast. Landscape in number of germs.

A fitness landscape is a geometry which maps inputs/strategies/ market factors to profitability. Each landscape has a peak which gives max returns. Standard economics assumes a U-shaped cost curve or globally concave (profits) function so landscape is simple: To get to bottom or top differentiate control variables,set them to 0 and solve. Negative second derivatives/matrices of second partials guarantee max or min. **But the world has "rugged" landscapes,** per Sewell Wright's landscapes to illuminate evolution. To find solution with complicated model, do **gradient search using a hill-climbing algorithm**.





Fitness landscapes With Gene A and B

	B	b
A	AB (1.10)	Ab (1.06)
a	aB (1.04)	ab (1.00)

A	AB (1.10)	Ab (0.90)
a	aB (0.95)	ab (1.00)

Why are landscapes for many decisions multi-peaked "rugged" so cannot search entire space to find the best:
1)Time to process the information is too limited to see all choices so better to make approximate decision – "bounded rationality". (But Alpha-Zero other programs and eventual Quantum Computer may do it all fast)
2) Too many choices → Choice Overload

Professor S, told me he could not buy groceries in the US because stores had too many options. His wife would send him to pick up cheese, bread, coffee, cereal. But how could he decide? There were 12 different cheeses, 10 brands of coffee, 17 breads, some packaged, some store baked, and 15 cereals… too many options (12 x 10x 17x 15) to find best. He returned home with nada. Easier in war-wrecked Poland where you buy whatever they had.

The Guardian, Wednesday 21 October 2015 "From jeans to dating partners and TV subscriptions to schools, we think the more choices we have the better. But too many options create anxiety and leave us less satisfied. Could one answer lie in a return to the state monopolies of old?" Cherneov, Bockenholt, Goodman, "Choice overload: A conceptual review and meta-analysis (Journal of Consumer Psychology, 2015)Pp 333-358 link attributes of decisions to indicators of choice overload –dissatisfaction, decision regret, deferral, switching over time.

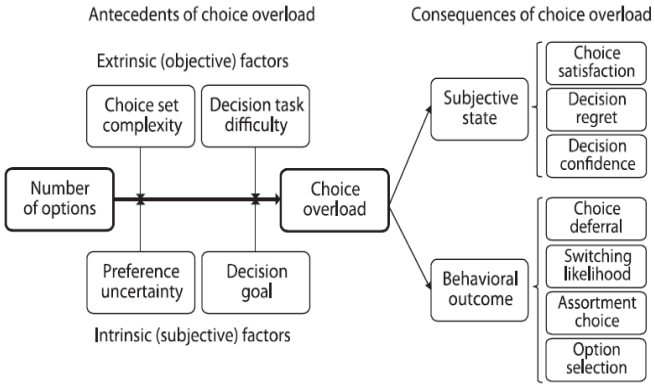


Table 2
A summary of the meta-analysis results.

Effect	Estimate	SE	T	P
Intercept (no moderators)	.41	.14	3.0	.01
Moderators of choice overload				
Choice set complexity	.55	.07	7.9	<.001
Decision task difficulty	.37	.08	4.7	<.001
Preference uncertainty	.32	.07	4.5	<.001
Decision goal	.56	.06	8.8	<.001
Measures of choice overload				

3. **Multiple equilibrium bcs theory is broad/weak.** Baumol-Gomory's Global Trade and Conflicting National Interest analyzes trade in "retainable industries" where scale economies/tech advantages mean the first country in industry dominates. With N goods and 2 countries and comparative advantage guaranteeing each country produces at least one good, you get 2^N - 2 equilibrium with some equilibrium good for one country but not the other.

Why 2^N - 2? T**he weak restriction of trade theory is that each country has at least one industry.** Absent the restriction there are 2^N possible assignments/combinations. Ruling out the 2 cases where country A or B have all industries leaves 2^N – 2. If N=3 one country could have industry A or B or C or A&B, B&C, A&C but the other country has to have at least one industry. This gives 6 possible equilibrium per, 2^N - 2 = 6 for N=3

The general formula is $\binom{N}{1} + \binom{N}{2} + \ldots + \binom{N}{N-1}$ which you can show is 2^N - 2

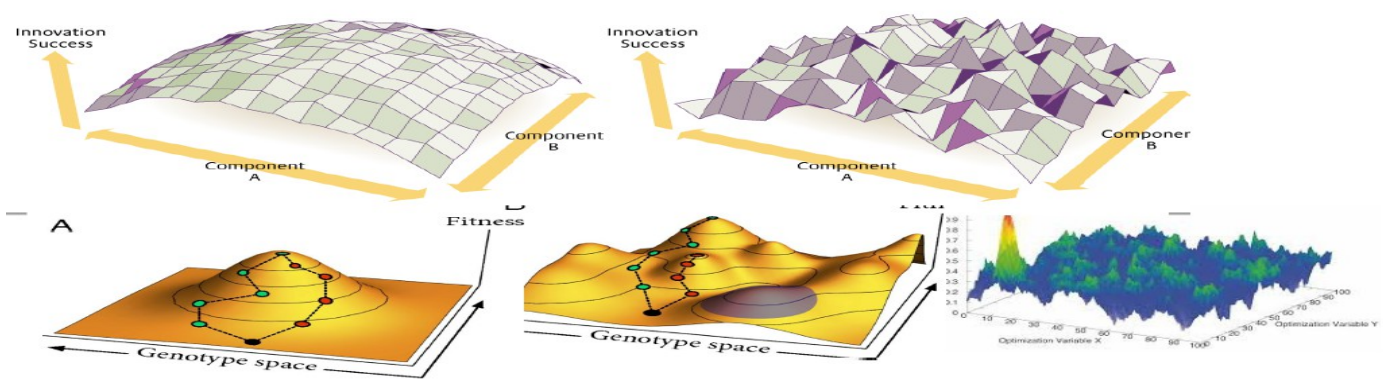
4)Landscapes change, so today's "perfect mate" turns into tomorrow's disaster.

A fitness landscape model is a representation of your theory/view of how to boil a set of outcomes into a single metric and of how easy or hard it is to change the factors that determine the maximand.

Measuring the shape of landscapes/search algorithms

To analyze landscape model, Need to specify **maximand**, which can be difficult, per the accounting decisions and protocols used to report profits; and uncertainty about preferences; **inputs to maximand** and a **distance or moving metric.** If the allowable move is a one-step, you get one landscape; if it is two-steps, the number of neighbors grows–> a different landscape.

One way to categorize a landscape is in terms of # of local optimum -- points where immediate neighbors give lower outcomes. The more local optimum, the more **rugged** is the landscape, which makes searching for the best difficult, especially if some high values are found in isolated areas with no high neighbors:



Search algorithms explore landscapes. On a concave landscape, **derivatives/gradient search** – hill-climbing.--works. Pick a point. Examine neighboring points -- incremental single attribute changes -- and follow steepest path up.

Example: The space of strategies is (a,b,c,d,e) where you choose 0/1 for each letter and the value at any point is the sum of the terms so the max is (1,1,1,1,1) with a value of 5. Pick a starting point -- (0,1,0,0,0). Look at neighbors and notice that (1,1,0,0,0) does better. Choose that point. Check again and again and you reach (1,1,1,1,1)

On a rugged landscape there is no best algorithm for all spaces. Can always devise a landscape where Algorithm B works better than A and conversely. If 4 is max, best estimate is 4, but 4 will not work if max of 30. Since uncertain about the landscape, seek a procedure that does well over many landscapes rather than one that best fiigureor the landscape you *think* you are on. Economics and math of search say "explore a lot before decision".

Kauffman's N-K Model

Consider three landscapes with N-input vectors and a geometry/moving rule that determines distance between vectors.

Random landscape randomly assigns profits to **all elements io** N-input vectors, so that there is no pattern and only a complete search finds global optimum. Example: Profits depend on 10 factors, each of which is 0/1. A point has N **near neighbors** that you reach by modifying one value. The N neighbors and the current point could potentially be a local maximum so in a n-hood each point has a 1/(N+1) chance of being a local maximum – with 10 factors, any combo has 1/11 chance of being a local max. There are 2^{10} (1/11) = 93 local max **unconnected** in the space. 000 0000 111 might be assigned 89; 100 0000 111 might be 6; 010 0000 111 might be 25; etc

Concave Landscape. Summing the value of **individual** inputs (0,1) to profit gives concave landscape. The max will be (111 1111 111) for 10. Nearest points will be 9 1s and one 0; 2nd nearest vectors 8 1s and two 0s.

Correlated Landscapes are in-between. Nearby points have similar values. Bundles of inputs that work together. Kauffman's **NK landscape** builds correlated landscape from simple model.

N = # factors (0/1, 0/1, 0/1, etc ... 0/1). Profit from each factor $w(s_n)$ Total profits are sum $W = \sum(w(s_n)$. If W was wins for a sports team W depends on individual production of each worker. **But interactions with other persons** on the team affects each person's output**.**

Let K = # of other players that affect the profitability of any player: $w(s_n)$ = f(Values of K other player). There are $\mathbf{2^{K+1}}$ combinations that determine profit contribution of each player: the 0/1 attributes of the player times the 2^K possible combinations of other players that impact that one. The K parameter builds in **non-linear interactions**. If there are 5 players (think basketball) and K=1, the profit contribution of anyone depends on each player and 1 other– say value of player e being 0 or 1 depends on a. The profit associated with a and e vary when you change **a** but the value of b, c, are unchanged. Scale the measure of so that profit of each unit is 0-1. Starting point is:

Attribute					Contribution to Profit					
a	b	c	d	e	a	b	c	d	e	
1	1	0	0	1	.6	**.3**	**.5**	**.2**	.7	for 2.3

Now change **a** from 0 to 1 and get changes in profitability of a, e, :

a	b	c	d	e	a	b	c	d	e	
0	1	0	0	1	.5	**.3**	**.5**	**.2**	.9	for 2.4

Switching **a** to 0 reduces its contribution to profits but raises the contribution of e. The landscape's structure comes from consistent contribution to profits of bcd. **The true MP of changing a from 1 to 0 is 0.1, not the -0.1 change in a's output.** Basketball's +/- statistic for net points while the player is on the court is a measure of that player's impact on the entire "profits" of team compared to another player being on: 1 could be player x; 0 some replacement player.

K tunes the ruggedness of the landscape.

K=0 is a concave landscape, since each *factor* affects profits independently.

K=N-1 gives a random landscape. The value of each *combination* is unrelated to any other. If you change an item, the value of ALL others can change. **Configurations are not neighbors.** The max change is N.

Example: N = 5, K = 4 (1,1,0,0,1) has profits (.5, .5, .6, .7, .2) --->2.5
(0,1,0,0,1) has profits (.3, .7, .9, .1, .3) --> 2.3.

K= 1 leaves the contribution to profits of 3 factors alone, K= 2 leaves contributions of 1 alone while K=3 affects all but one factor, so you have a stable base of (N-K)/N percent of your value function. Configurations that give big/little values are neighbors so you can't decentralize search for global optimum into set of separable choices. This makes ORGANIZATION more important.

As K rises from 0 to N-1, you get more local peaks and the height of average peak falls –> loss of potential incremental gain. The chance of an isolated global peak increases.

II. Propositions and Applications

1) Going slow can be best (David Kane): Take a quadratic profits function, $\Pi = \sum aX + bX^2 + c\, XiXj$, where a,b,c are randomly selected over the line -1 to 1. The moving rule is to shift $ to K other inputs. Consider 3 search algorithms:

STEEPEST ASCENT: Check your neighbors one flip away: if you are 00000, look at 10000, 01000, 00100, 00010 and 00001. Choose the one with the highest value. Repeat and rush up hill fast as you can – works if concave.

NEXT ASCENT: Pick first neighbor that improves your score. If you are 0000 you look at 1000. If 1000

improves score, choose it and repeat. If 1000 does not help, look at 0100. If this improves your score choose it and start again. If not, look at 0010. This widens # points you look at. You go up with smaller positive gain.

RANDOM MUTATION ASCENT. Randomly flip a locus and whenever you get a gain, go there. .

. MEDIAN ASCENT -- pick the midpoint of gainers in n-hood; LEAST ASCENT -- pick smallest gain in n-hood.

Strategy	1	2	3	4	5
Steepest Ascent	15.6 (0.2)	24.7 (0.3)	32.1 (0.3)	37.9 (0.4)	43.0 (0.4)
Median Ascent	16.5 (0.2)	25.3 (0.3)	32.2 (0.3)	39.0 (0.4)	43.7 (0.5)
Least Ascent	19.1 (0.2)	30.1 (0.3)	38.4 (0.4)	43.1 (0.5)	48.8 (0.6)

Table 1: Mean normalized profits over 1,000 landscapes for different firm strategies and connections per input. Coefficients for the profit function are drawn from [−1, 1]. Standard errors in parentheses.

Who does best? LEAST ASCENT. Who does worst? Steepest Ascent. Why? With many local maximum going up quickly risks getting stuck at local max. Going slowly **explores the landscape more.**

Nahum et al PNAS, June 2015 : "A tortoise–hare pattern seen in adapting structured and unstructured populations suggests a rugged fitness landscape in bacteria" argues that since on single-peaked landscape, all uphill paths converge, there is less search and fitness converges to average quickly while on rugged landscape, *which could involve evolutionary changes on different islands*, creatures develop higher fitness with more mutation variation. They find bacteria reach higher fitness and accumulate more mutations under restricted migration than unrestricted migration, which is consistent with a rugged topography.

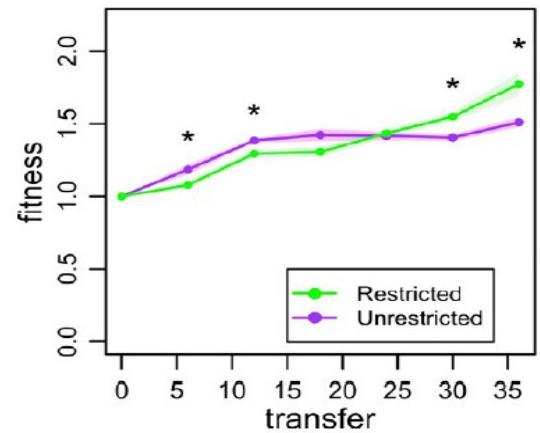
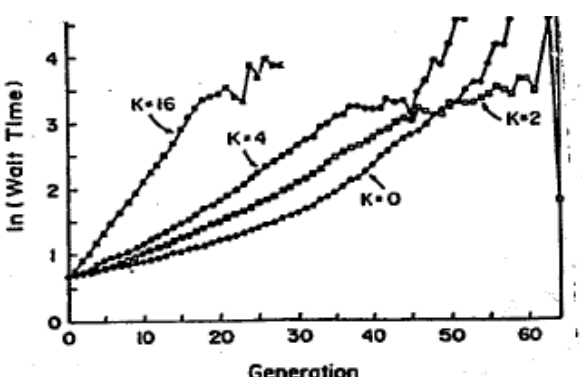


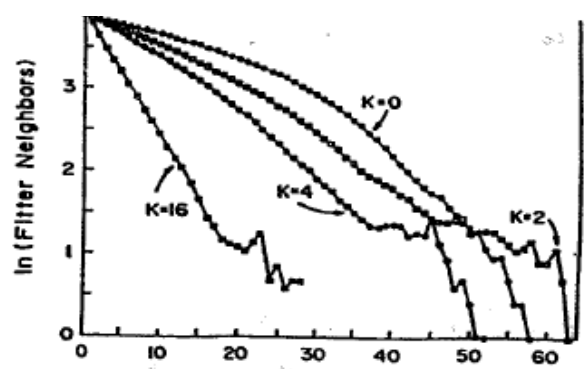
Fig. 4. Bacterial fitness. Metapopulations of bacteria evolved where mi gration was spatially restricted or unrestricted. The average relative fitnes

2) If K is large, system is highly connected and many elements must change to move to higher peak –> discontinuous change. Improvement takes longer – big change is infrequent.

After fall of Soviet Union some economists recommended "big bang" move to markets which generally failed; while China's more gradual move from state-planning to markets spurred huge growth. Industrial Relations focuses on the landscape for HIGH PERFORMANCE workplaces defined by a configuration of attributes: job rotation; training, employee involvement committees, employee ownership. Changing one thing might do little good: in connected system/ correlated landscape derivatives of single attribute varies with the value of other variables. With more connections -- larger K -- there are fewer fitter neighbors and it takes longer to find them: Graphs show a) (ln) time it takes to find better neighbor (which rises with K) and b)# of fitter neighbors you find over time is smaller the larger K.



Uniform Distribution N=96



Discontinuous changes shows up with long periods with no improvement followed by burst of gains because many changes are needed to improve highly connected system. Say the chance a factor changes is 50% per year and you need 5 changes to improve. The probability that all 5 changes occur in a year is $0.5^5 = 0.03125$, which indicates that likelihood of big change is once in 32 years. With small K, two changes could produce frequent gradual advances.

Clinical Trials Simulation (Eppstein, Horbar, Buzas, Kauffman, 2012) compares RCT, where practice passes statistical test to "quality improvement collaborative (QIC)", which accepts changes that work in neighbors. "search strategy modeled after QICs yields robust improvement in simulated patient outcomes". QIC agents adopt practice that yields higher survival at their locale, *without regard to statistical significance,* and "greater improvements in health outcomes … than RCTs." Why? "Due to a combination of … accepting things that work but do not meet significance due to small

n) and increased ability for agents to respond differently in different local contexts" (ease finding local optimum). This makes sense – you should choose positive effect medicine X if your life depended on it regardless of stat significance.

Simulated Annealing (the value of going in the wrong direction)

Go down before you go up to escape local optimum on rugged landscape. You check out potential mates at your high school and pick the best for you. But **maybe** there are more compatible folks at college. The only way to escape is to go the wrong way: drop Mr/Ms HS compatible and search anew.

Simulated annealing is a "technique to find a good solution to an *optimization problem* by trying random variations of current solution. A worse variation is accepted as new solution with a probability that decreases as the computation proceeds. The slower the *cooling schedule*, or rate of decrease, the more likely the algorithm will find an optimal or near-*optimal solution*." (US National Institute of Standards)

Say you want to minimize cost. Start at S with f(S). Pick new point s depending on the prob: P(s) = exp (-δ/t for temperature) -- probability of going in higher cost direction f(s) - f(S) > 0. Example: you believe efficiency wage theory and raise pay to motivate workers to work harder which raises costs in short run but could reduce costs over long run. Since δ = f(s) - f(S) >0 it measures willingness to go the wrong way. Bigger t --> more willing to go wrong way. If δ is large, you get a big increase in cost and P(s) is smaller than if δ is small. Move in wrong direction when landscape is flat and you leave local optimum but are unlikely to find better outcome.

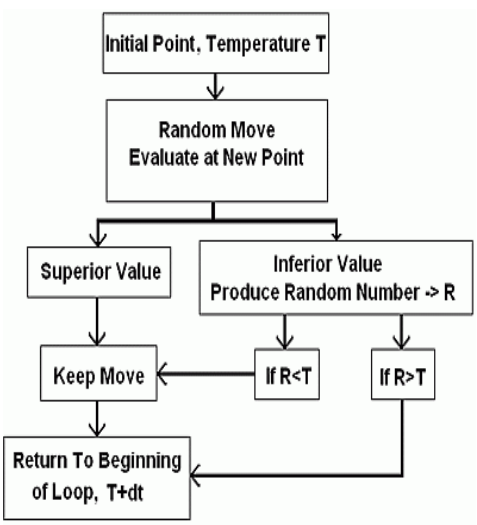
As t increase, P--> 1 for positive deviation, so you are more willing to go in the wrong way

Example: t =5 δ= 10, then P(s) = exp (-2) = .14 t= 10 δ = 10, then P(s) = exp (-1) = .37

t =5 δ = 25, then P(s) = exp (-5) = .007 t= 10 δ = 25, then P(s) = exp(-2.5) = .08

Reading across, as t rises, P(s) goes up. Reading down, as δ goes up, P(s) falls. Algorithm starts with a small t that searches widely to find potential peaks, then narrows in on the best one. **t** determines how much Simulated Annealing explores new areas vs areas of known profitability. Since algorithm always accepts a new point that is better, you will approach local optimum. Accepting move in the other direction allows you to escape from local optima. The algorithm makes no assumptions about function to be optimized, making it robust across many landscapes.

Here are two descriptions of the algorithm, where a(t) reduces t over time and your willingness to search widely.



Select an initial solution s_0;
Select an initial temperature $t_0 > 0$;
Select a temperature reduction function α;
Repeat
  Repeat
    Randomly select $s \in N(s_0)$;
    $\delta = f(s) - f(s_0)$;
    If $\delta < 0$
    then $s_0 = s$
    else
    generate random x uniformly in the range (0, 1);
      if $x < \exp(-\delta/t)$ then $s_0 = s$;
  Until *iteration_count* = *nrep*
  Set $t = \alpha(t)$;
ntil stopping condition = true.
) is the approximation to the optimal solution.

Simulated annealing works on problems with solutions and comes close o the optimum. It works on traveling salesman http://www.math.uu.nl/people/beukers/anneal/anneal.html gives an applet for this. You pick cities on a grid and the annealing program searches for the minimum distance to travel to reach all of them. Mathematica implements Simulated annealing http://www.sph.umich.edu/csg/abecasis/class/2006/615.19.pdf. Sebastian Herrmann Complex Network Analysis of Fitness Landscapes Dissertation zur Erlangung des Grades eines Doktors Universit at Mainz Jahre 2016: "For each problem … we performed 1,000 independent runs of hill climbing and simulated annealing (both with random initial solutions).We measured search performance by 1. the success rate (ps), which is defined as the percentage of runs that find the global optimum, and 2. the average number of fitness evaluations. We assessed the quality of the different predictor metrics by the R2 of a linear regression of outcome on predictor.

GENETIC ALGORITHM evolves a population of strategies into a new population with higher average profitability using **trial and error improvements** and **reproduction/survival of the fittest via Proportional Fitness Reproduction (PFR)** to search for best outcome. To do a GA search you: CODE strategies as 0/1 for different attributes.

Determine profits of current strategies

Create new population using some form of PFR. This is the *mating population*

Use "genetic operators" to create new strategies by :

cross-over, take two strings -- **101**101 and 001**001**; split and join to get: **101001** and 001101.
random mutation, switch a random 0/1 or randomly change each point by some probability.
Inversion pick two points and switch them **10**001 becomes **01**001

EXAMPLE: Choosing research paper for course, you decide who to work with, data to use, modes of analysis, time you will give to analysis vs writing. You have data on all past papers in course and make your plan on the basis of their success. Which mix of attributes will make your paper s star? Data on three attributes of previous papers show:

Alone or with friend		Emp/Theory	Time on Analyst	Binary Rep	Profits
1	Friend	Empirical	Mostly writing	011	3
2	Friend	Theory	Mostly writing	001	1
3	**Alone**	**Empirical**	**Mostly Analysis**	**110**	**6**
4	Friend	Empirical	Mostly Analysis	010	2

Why is 110 so profitable? Here are all possible explanations consistent with data

It's working alone	1**, where * refers to any value/doesn't matter.
It's empirical	*1*
It's no endorsement	**0
It's working alone and empirical	11*
It's working alone and no endorsement	1*0
It's empirical and no endorsement	*10
It's alone, empirical, and no endorsement	110
Nothing matters	***

These are **schema/schemata,** which generalize strings in terms of a common property, * for the generalized property:

*11 is the schemata for 011 and 111. It's: anything 11
1 is the schemata for 110, 010, 011, 110. It's anything 1 anything.

Here are the average fitness for all of the 8 schemata to which 110 belongs:

Table 3.13 The $2^L = 8$ schemata to which individual 110 belongs.

	Schema	Average fitness
1	110	6
2	11*	6
3	1*0	6
4	1**	6
5	*10	4
6	*1*	3.67
7	**0	4
8	***	3

ne

schema to which it belongs. 001 provides information on 8 schema. When we have lots of observations on a schema, we have better information about whether high profits lie in its area than when we have few observations. Implicitly each schema has a FITNESS VALUE -- the average of the value of its strings. Since each observation– 110, 001, 011, 010-- belongs to 8 schema, the 4 observations give LOTS of information on the space of outcomes "spanned" by the schema. GA converges fast to profitable strategies because each observation contributes info about many parts of the space of potential outcomes.

Table 3.11 Individual strings belonging to each of the 27 schemata.

	Schema	Individual strings
1	000	000
2	001	001
3	00*	000, 001
4	010	010
5	011	011
6	01*	010, 011
7	0*0	000, 010
8	0*1	001, 011
9	0**	000, 001, 010, 011
10	100	100
11	101	101
12	10*	100, 101
13	110	110
14	111	111
15	11*	110, 111
16	1*0	100, 110
17	1*1	101, 111
18	1**	100, 101, 110, 111
19	*00	000, 100
20	*01	001, 101
21	*0*	000, 001, 100, 101
22	*10	010, 110
23	*11	011, 111
24	*1*	010, 011, 110, 111
25	**0	000, 010, 100, 110
26	**1	001, 011, 101, 111
27	***	000, 001, 010, 011, 100, 101, 110, 111

A schemata is a theory of how the world works. Take the theory "1**, then profits rise." If 0s and 1s arrive randomly, 1** will occur in 1/2th of the cases. It won't require too many cases to determine if it is right. A few 0s in first spot and high profits will kill 1** off. A more specific rule would be if 111. This occurs in 1/8th of the cases. It will be harder to see if it is right. **Easier to learn from general rules than from specific ones. The # of schemata > # possible data points**. Problem has 8 possible observables (2^3) but $3^3 = 27$ schemata, since 3 symbols for each bit in the string

instead of 2. The GA gains power because e**ach string gives information about many schemata.** Observations about "other data points" give clues about why 110 is profitable. If 0 ** has profits you know 1 in first space cannot be **necessary.** Maybe 0 is just as good or maybe * nothing matters. A single string with 3 bytes belongs to 8 (2^3) schemata. Most profitable 110, belongs to the 8 schemata listed above. In general string with L bytes belongs to 2^L schemata.

Why not pick the most profitable string and stick with it? Because you learn nothing about the landscape. If must decide which strategy today, choose 110. But further search can reveal other ways of making profit: 111, 100,101, 000. Economics says we should compare costs to benefits of search. The cost of experimenting is the opportunity cost of selecting current best: what we make on others vs what we would make with 110:

for 111, we make 7, so the "cost" is 7-6 =1 (it's a benefit, not a cost)
for 100, we make 4, so the cost is 4-6= -2
for 101, we make 5, so the cost is 5-6 = -1
for 000, we make 0, so the cost is 0-6= -6

If we randomly select the possibilities to explore, the cost would average -2. **But if we direct our search to more profitable areas, the cost would be smaller: GA searches in areas where we expect more profitable outcomes.** As long as global optimum is near areas of high probability, GA would direct us to the right areas since it "infers" that good profits come from strings like 110. If the global max was 000, we would go in the wrong direction.

In two armed bandit problem there are two slot machines, each has a payoff with a variance v. The optimal solution is not to pick the one that has the highest value after your first "experiment" nor the one with highest value even after a number of tries. Always check the lower paying one -- that you simply had bad luck on it previously. The optimal allocation gives proportionately more trials to the higher-paying arm, with the proportions dependent on variances. If environment can change best strategy at t' may differ from the one at t so keep exploring.

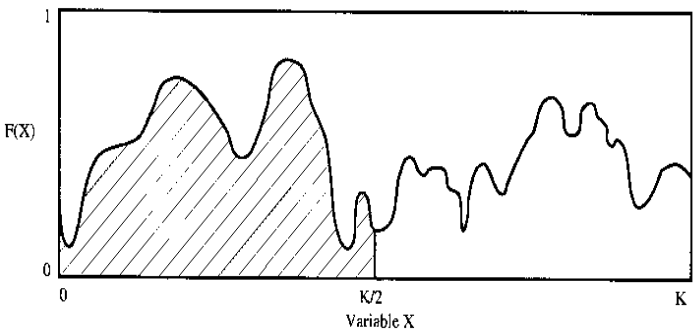
The r**elative fitness/profits** of a string determines the **probability** that it enters the next generation. Total profits/fitness is 12, the worst is 1 and the best is 6. Average profits are 3. A string with a score of 6 would twice as much as chance of being in the next generation as a string with 3: its probability of being selected is 0.5. If we pick four strings, a string with 0.5 fitness would very likely (94%) be selected at least once (prob you will not be selected is $0.5^4 = .06$). A string with .05 fitness would have a probability of NOT being selected at all of $.95^4 = 81\%$

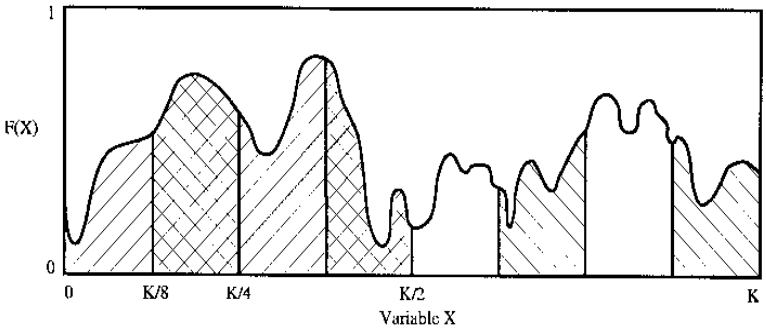
Drawing from the probability urn differentiates this procedure from the computer tournament, where we stock the next generation with the relevant **proportions**. The probabilistic approach increases the chance that strategy with Probabilistic structure allows low return to persist. This adds diversity into the population. Allowing some low return strategy to persist is a common way to try to keep away from local optimum, as in simulated annealing.

SHORT you-tube TUTORIAL videos -- 4-10 minutes. https://www.youtube.com/watch?v=Y-XMh-iw07w From IIT India https://www.youtube.com/watch?v=Z_8MpZeMdD4 and https://www.youtube.com/watch?v=ra13Sv7XZ3M http://math.hws.edu/xJava/GA/ has a java applet window showing the trend upward with lots of variation in GA search. Whitley, Darell is good; www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf

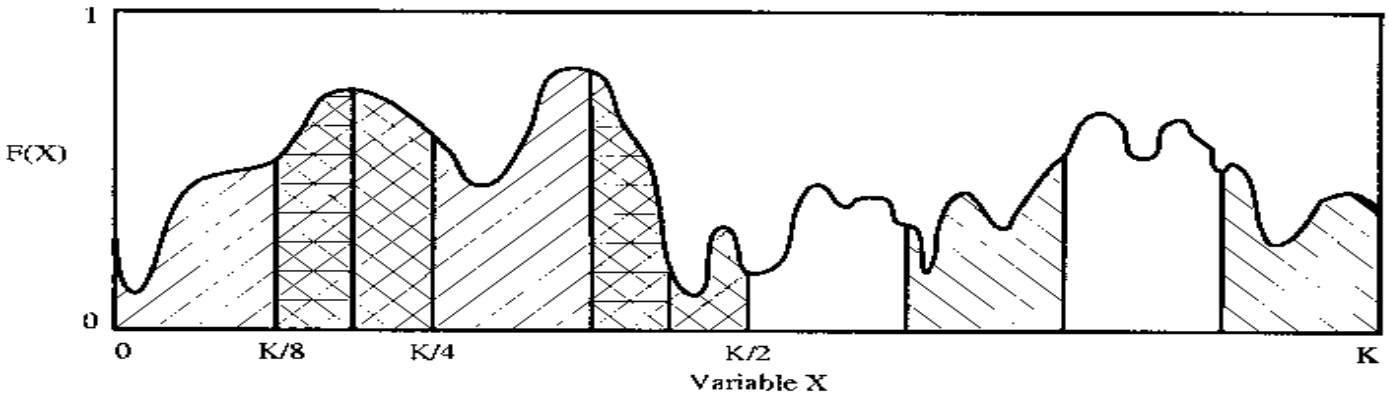
The Fundamental Theorem of GA (Wasserman, p 87) says that GA using fitness-proportionate reproduction and cross-over and mutation gives proportionate growth to more fit schemas.

GA works by **partitioning variable space into areas of higher and lower profitability** and focusing search in the higher profitability/fitness areas. Left graph partitions space by first element, 0 to left and 1 to right. If strings in 0** partition /// have higher average profit than those in the 1** area search the hatched area. Say 0 in 2nd position has higher average value, denoted by \\\. Take the two and the algorithm says search in double-hatched areas.





Now take a third point where XXX denotes profitable areas. Most of your search would be in the two disjoint areas that combine the profitable places for the three "search mes".



The GA divides the space into areas where the three schemata (0**, ***0* and **0**) **say are promising. Largely unaffected by local optima since the partition is for above average outcomes.** The extent to which this damages a profitable strategy depends on the the **defining length** of the string, d, which is the distance between furthest specific (0 or 1) positions –the nearness of the well-specified parts. The 7 element schema 1**0*** has defining length 3 because you move 3 spaces to get from 1 to 0; whereas *001*00 has defining length 5 and ** 011** has defining length 2.

Problems with GA: the Royal Road Clunker:GA is supposed to work well because it samples spaces efficiently by combining goodbuilding blocks. Obtain small groups that work well together and combine them.

The Royal Road Function is a seemingly perfect set-up for the GA. The maximum is 1111 1111 1111 a combination of three blocks of four units:

A = 1111 **** **** B = **** 1111 **** C = **** **** 1111

Since GA lives on connecting correlated bundles, analysts expected it to do great on this problem. Mitchell, Forest and Holland compared GA with 3 types of hill-climbing over this landscape: SA steepest ascent -- check all Ns and pick biggest; NA -- nearest ascent -- choose first point with increase; RM -- random mutation -- random flip of bit, choose if works best. Here are results:

Table 4.1 Mean and median number of function evaluations to find the optimum string over 200 runs of the GA and of various hill-climbing algorithms on R_1. The standard error ($\sigma/\sqrt{\text{number of runs}}$) is given in parentheses.

200 runs	GA	SAHC	NAHC	RMHC
Mean	61,334 (2304)	> 256,000 (0)	> 256,000 (0)	6179 (186)
Median	54,208	> 256,000	> 256,000	5775

ON THE ROYAL ROAD, random mutation beats GA! It takes fewer searches/less time to reach the optimal.

What goes wrong? GA GETS CAUGHT ON LOCAL MAX because you are not sampling independently in each region. By chance you get A at the beginning, then you will get lots of AB'C' . You need mutation to get out of local maximum. By contrast RM does a slow (tortoise) search over the entire space. GA focuses search in areas where by chance you get an early good return.

If you are searching over the space of dates, and you go out with people in real time, searching for the perfect person could leave you unhappy most of your life. The algorithm that yields (0.7, 0.5, 0.8, 0.6, **0.6**), is inferior to an algorithm (0.1, 0.2, 0.3, 0.1, **1.0**) if the goal is the perfect 1.0 . But if you are 92 when you meet perfect 1.0, the total score over the search period is 3.2 for the first search vs 1.7 for the 2nd. At reasonable discount and potential life span, you are unlikely to reach the future 1.0s that make search for perfection the best strategy.