The Evolution of Complexity seen as a problem of Search:

Can we predict a priori which search algorithm will work best on which problem?

Chris Stephens, C3 y ICN, UNAM

CCS 17, Cancun, Mexico

17-20th September 2017

Isn't Everything just Search?

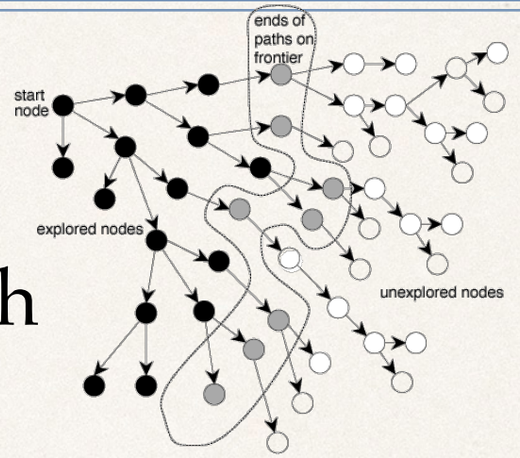
- Need a search space
  - Search points/configurations
- Need a function to qualify the search
  - Objective function/fitness/"success" measure

Need a problem to "solve"

- Need a search algorithm
  - Search Heuristic - EAs/Evolution/Nucleosynthesis/…

Need a way to "solve" it

- Exogenous versus endogenous
  - Objective function/fitness/success measure
  - Stopping criterion
  - Natural (Physical)/Artificial (Mathematical)



Algorithm 2 Algorithmic frame for a local search algorithm

```
1: N := number of repetitions;
2: s̃ := ∅;
3: for i = 1 to N do
4:    s := initial solution;
5:    while there is a neighbor of s with better quality do
6:       s := one arbitrary neighbor of s with better quality;
7:    end while
8:    if s is better than s̃ then
9:       s̃ := s;
10:   end if
11: end for
12: return s̃;
```

Integrating equations of motion

$$F_{i,x} = m_i \frac{dv_{i,x}}{dt} \quad F_{i,y} = m_i \frac{dv_{i,y}}{dt} \quad F_{i,z} = m_i \frac{dv_{i,z}}{dt}$$

$$\frac{dv_{i,x}}{dt} = \frac{d^2 x_i}{dt^2} \approx \frac{x_i(n+1) - 2x_i(n) + x_i(n-1)}{\Delta t^2}$$

This works out to give the Verlet algorithm,

$$x_i(n+1) = 2x_i(n) - x_i(n-1) + \frac{F_{i,x}}{m_i}\Delta t^2$$

There are a lot of problems and a lot of search algorithms!

No Free Lunch Theorems

1 Wolpert, D.H., Macready, W.G. (1995), No Free Lunch Theorems for Search, Technical Report SFI-TR-95-02-010 (Santa Fe Institute).

2 Wolpert, D.H., Macready, W.G. (1997), "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation* **1**, 67.

Over all problems to solve, no way of solving them is any better than any other

- What features of a given problem can tell us which search algorithm to use?
- Which search algorithms work best on which problems?
- What problems are "special"?
- Is there anything special about "real world" (physics, biology,…) problems and/or search algorithms?
- Which search algorithms work best on these "special" problems?

Searching for the magic bullet search algorithm…

Table III. Error

Data	AODE	NB	Bag ODE	Bag ODE	LBR	TAN	SP-TAN	J48	Boost J48
adult	0.152	0.168	0.172	0.170	0.140	0.147	0.147	0.146	0.169
anneal	0.065	0.082	0.064	0.059	0.064	0.067	0.067	0.157	0.101
balance-scale	0.302	0.303	0.310	0.267	0.302	0.303	0.300	0.274	0.238
bcw	0.027	0.030	0.038	0.036	0.030	0.050	0.030	0.087	0.048
bupa	0.424	0.424	0.424	0.426	0.424	0.424	0.424	0.428	0.395
chess	0.140	0.143	0.144	0.146	0.141	0.128	0.137	0.144	0.124
cleveland	0.176	0.174	0.175	0.170	0.174	0.176	0.178	0.260	0.227
crx	0.163	0.171	0.165	0.162	0.172	0.177	0.172	0.172	0.179
echocardiogram	0.382	0.389	0.382	0.364	0.392	0.388	0.388	0.372	0.366
german	0.262	0.268	0.268	0.262	0.269	0.277	0.268	0.296	0.291
glass	0.299	0.300	0.299	0.275	0.303	0.300	0.295	0.288	0.257
heart	0.216	0.215	0.217	0.201	0.215	0.236	0.218	0.269	0.254
hepatitis	0.140	0.139	0.137	0.129	0.140	0.143	0.138	0.173	0.163
horse-colic	0.219	0.221	0.227	0.217	0.210	0.213	0.219	0.226	0.229
house-votes-84	0.054	0.086	0.089	0.087	0.069	0.068	0.082	0.040	0.044
hungarian	0.173	0.169	0.173	0.177	0.173	0.179	0.172	0.211	0.212
hypothyroid	0.021	0.024	0.025	0.025	0.016	0.025	0.018	0.013	0.015
ionosphere	0.102	0.119	0.122	0.102	0.119	0.099	0.118	0.166	0.143
iris	0.058	0.058	0.058	0.054	0.058	0.056	0.058	0.060	0.059
labor-neg	0.150	0.150	0.150	0.135	0.196	0.168	0.154	0.239	0.192
led	0.258	0.255	0.268	0.270	0.257	0.271	0.259	0.318	0.318
letter-recognition	0.193	0.292	0.266	0.259	0.220	0.212	0.210	0.208	0.103
lung-cancer	0.556	0.556	0.556	0.540	0.557	0.562	0.555	0.616	0.608
mfeat-mor	0.311	0.317	0.321	0.312	0.313	0.312	0.314	0.300	0.305
new-thyroid	0.074	0.074	0.084	0.088	0.074	0.077	0.075	0.119	0.093
pendigits	0.037	0.132	0.067	0.059	0.065	0.066	0.055	0.065	0.021
post-operative	0.366	0.366	0.366	0.360	0.364	0.383	0.386	0.317	0.416
promoters	0.130	0.130	0.130	0.146	0.132	0.315	0.134	0.247	0.208
ptn	0.572	0.559	0.581	0.584	0.571	0.593	0.571	0.635	0.635
satellite	0.120	0.178	0.164	0.152	0.148	0.128	0.155	0.164	0.119
segment	0.071	0.112	0.116	0.094	0.092	0.082	0.090	0.065	0.041
sign	0.302	0.362	0.295	0.290	0.280	0.292	0.297	0.206	0.175
sonar	0.275	0.274	0.277	0.261	0.274	0.293	0.279	0.316	0.269
syncon	0.059	0.069	0.086	0.060	0.069	0.058	0.069	0.191	0.106
ttt	0.261	0.296	0.295	0.295	0.291	0.294	0.295	0.240	0.147
vehicle	0.383	0.444	0.438	0.406	0.385	0.382	0.428	0.334	0.277
wine	0.042	0.040	0.042	0.029	0.040	0.053	0.040	0.143	0.094
Mean	0.204	0.219	0.216	0.207	0.209	0.216	0.211	0.230	0.206
Geo mean ratio		1.124	1.115	1.048	1.049	1.102	1.056	1.225	1.026

Webb, Boughton and Wang, Machine Learning 2005

- 37 problems with 9 ''search'' (classification) algorithms
- No one works best on all, or even a significant subset
- Performance differences between algorithms of up to a factor of 3, on a given problem!

When and why does a particular search algorithm work best on a given problem?

Can we predict it?

Let's start with one "search" algorithm - the Naive Bayes classifier

What features of a given problem can tell us which search algorithm to use?

When is the Naive Bayes approximation not so naive?

Why the Naive Bayes approximation is not as Naive as it appears, Stephens, C.R, Huerta, H.F. and Linares, Ana Ruiz, Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on, pp. 1-6, (2015).IEEE.

Stephens, C.R., Huerta, H.F. & Linares, A.R. Mach Learn (2017). https://doi.org/10.1007/s10994-017-5658-0

$$P_{NB}(C|\mathbf{X}) = \frac{\prod_{i=1}^{N} P(X_i|C)P(C)}{(\prod_{i=1}^{N} P(X_i|C)P(C) + P(\mathbf{X}|\bar{C})P(\bar{C}))}$$

$$S_{NB}(\mathbf{X}) = \log\frac{P(C)}{P(\bar{C})} + \sum_{i=1}^{N} \log\frac{P(X_i|C)}{P(X_i|\bar{C})}$$

Constructs P(C | **X**) neglecting correlations between the Xi

Works pretty well on many problems. Why?

$$P_{GB}(C|\mathbf{X}) = P(C|\xi^{(i)}) = \frac{\prod_{\alpha=1}^{N_{\xi^{(i)}}^{C}} P(\xi^{\alpha}|C)P(C)}{(\prod_{\alpha=1}^{N_{\xi^{(i)}}^{C}} P(\xi^{\alpha}|C)P(C) + \prod_{\alpha=1}^{N_{\xi^{(j)}}^{C}} P(\xi^{\alpha}|\bar{C})P(\bar{C}))}$$

$$S_{GB}(\mathbf{X}) = \ln\frac{P(C)}{P(\bar{C})} + \sum_{\alpha=1}^{N_{\xi^{(i)}}^{C}} S^{C}(\xi^{\alpha}) - \sum_{\alpha=1}^{N_{\xi^{(j)}}^{C}} S^{C}(\xi^{\alpha})$$

$$S^{C}(\xi^{(i)}) = \sum_{\alpha=1}^{N_{\xi^{(i)}}^{C}} \ln P(\xi^{\alpha}|C) \qquad S^{C}(\xi^{(j)}) = \sum_{\alpha=1}^{N_{\xi^{(j)}}^{C}} \ln P(\xi^{\alpha}|\bar{C})$$

Let's embed it in a larger set of search algorithms where the factorisation of the likelihood is not maximal.

Generalised Naive Bayes

When is the Naive Bayes approximation not so naive?

We'd expect the NBA to breakdown when there are strong correlations… wouldn't we? i.e., for any problem with strong correlations we should choose another algorithm. But…

i) How we do quantify strong correlations?

ii) How do we turn that into a decision about which algorithm to use?

"What features of a given problem can tell us which search algorithm to use?"

$$\delta(\xi|C) = P(\xi|C) - P_{NB}(\xi|C) = P(\xi|C) - \prod_{i=1}^{m} P(\xi_i|C)$$

$$\sum_{\xi} \delta(\xi|C) = \sum_{\xi} \delta(\xi|\bar{C}) = \sum_{\xi} \delta(\xi) = 0$$

$$\delta(X_1X_2|C) = P(X_1X_2|C) - P_{NB}(X_1X_2|C)$$

"Large" values of $\delta(\xi|C)$ tell us that those features in ξ should not be separated… "Let no man…"

The $\delta(\xi|C)$ can't all have the same sign. Errors must cancel!

ξ is a "building block" of features

Same as linkage in population genetics - gives information on epistasis

When is the Naive Bayes approximation not so naive?

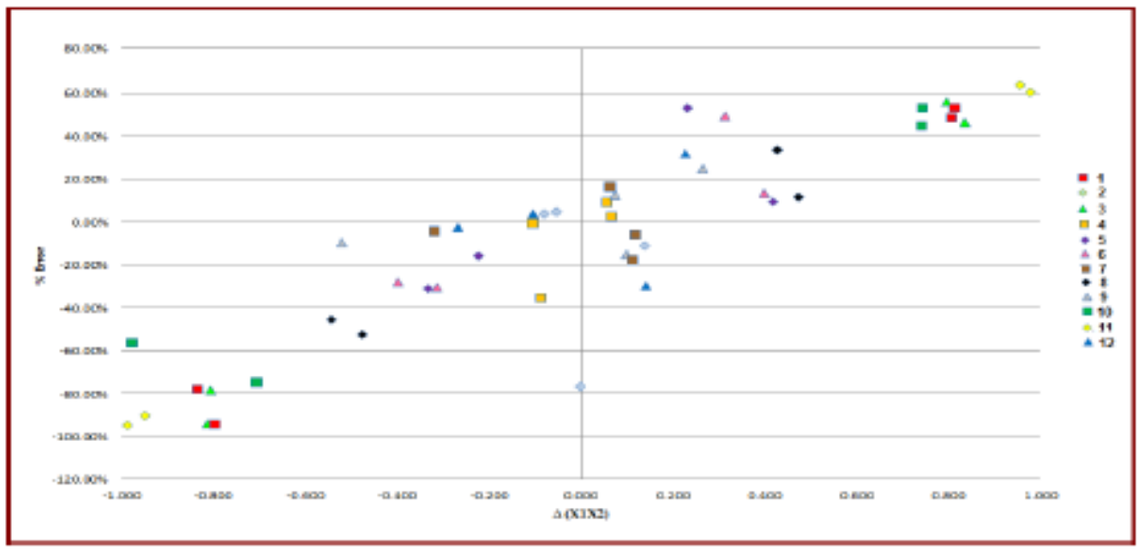


Fig. 5.1 Graph of % error in NBA of the posterior probability $P(C|X_1X_2)$ as a function of $\Delta(X_1X_2)$ for each feature combination of the 12 probability distributions of Appendix A

Make up some artificial two-feature problems where we can tune the degree of correlation

Our diagnostic predicts the performance of the NBA.

Extend this to 4, 6 and 8-feature problems by concatenating the 2-feature ones

Our diagnostic predicts the performance of the NBA.

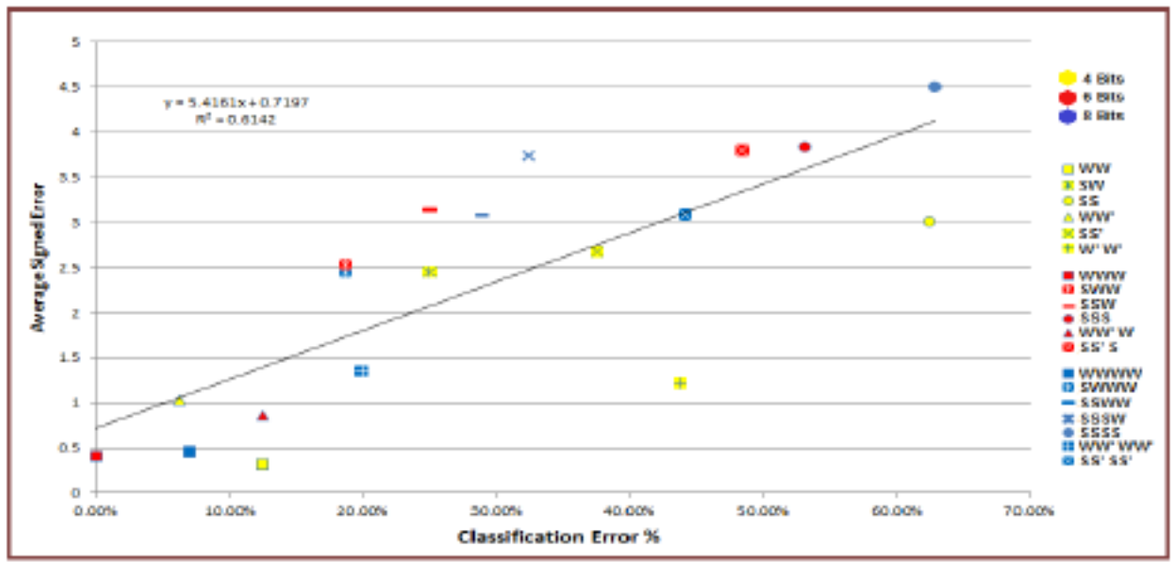


Fig. 8.2 Graph of Classification error vs. ΔS_{total} for different 4, 6 and 8 feature distributions.

Which search algorithms work best on which problems?

When is the Naive Bayes approximation not so naive?

Now let's go to "real" world problems - 20 UCI datasets

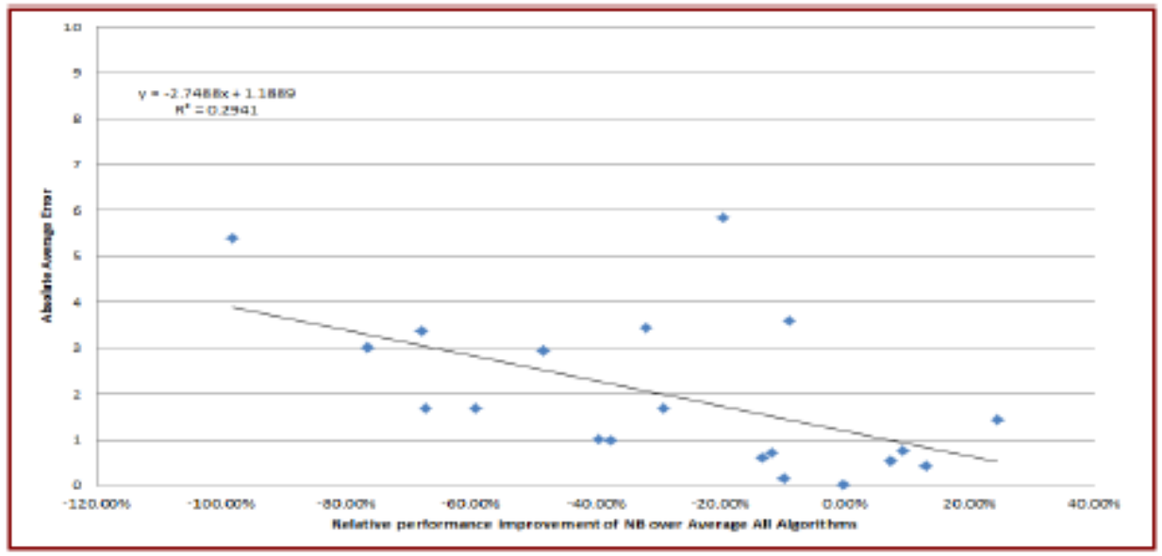


Fig. 9.1 Relation between the percentage relative difference in error between the NBA and GBA and the average absolute error for the 20 UCI data bases and averaged over the 5 GBA classifiers.

Our diagnostic predicts the performance of the NBA relative to more sophisticated generalisations

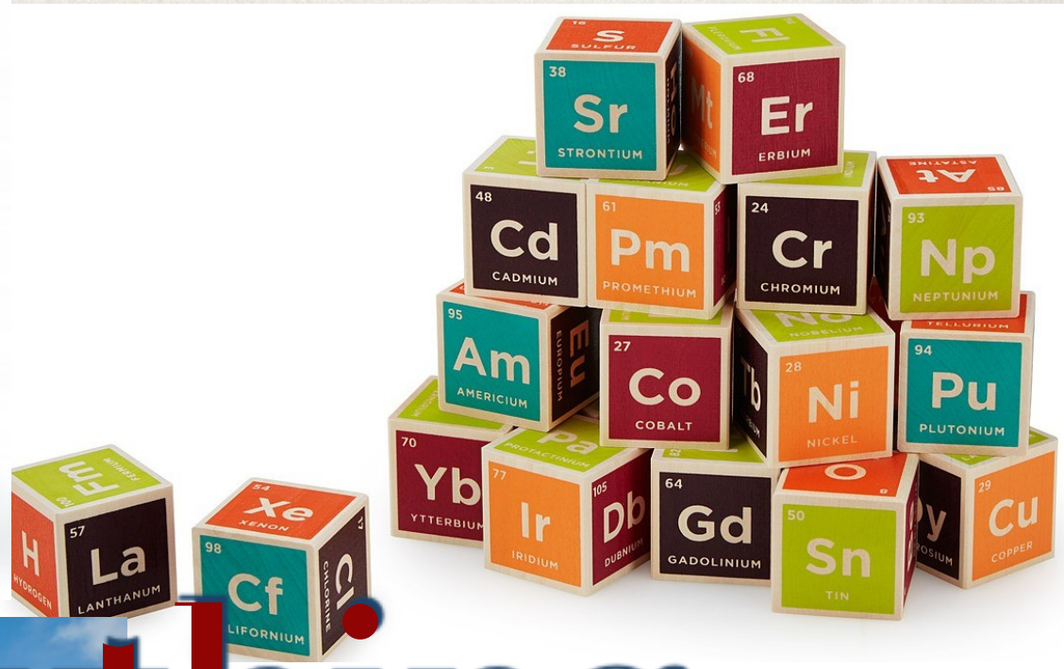
It's remarkable that one single diagnostic can give such good results on real world problems that are so very different.

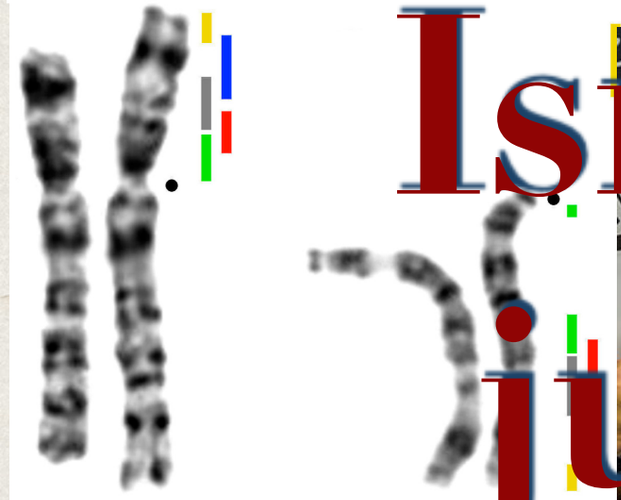
Meta-prediction algorithm - predicting which predictor will predict best

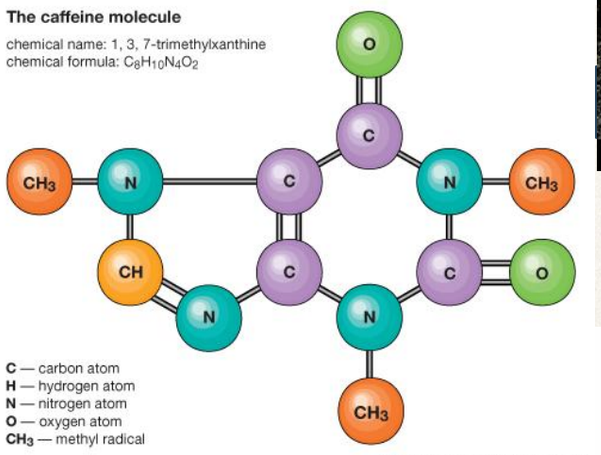
Classifier	Correlation Coefficient	Correlation Coefficient (no ionosphere)
GNB_s UCI	-0.45	-0.65
GNB_a UCI	-0.47	-0.68
AODE UCI	-0.61	-0.77
WAODE UCI	-0.53	-0.59
HNB UCI	-0.52	-0.62
Avg All	-0.54	-0.68
NBC Artificial	0.78	NA

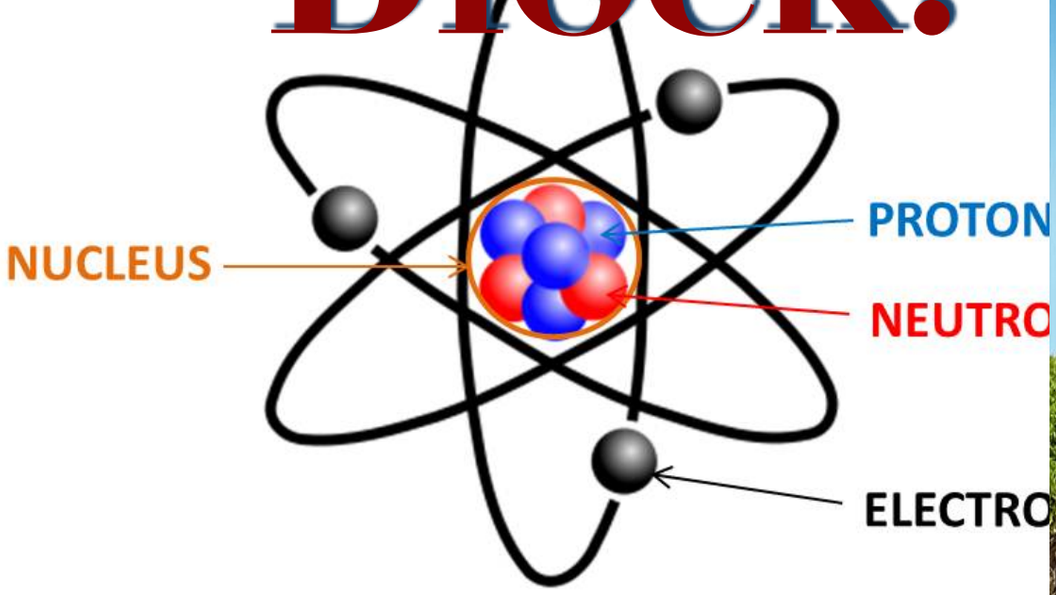
Table 9.3 Pearson correlation coefficients between the average of the absolute error $|\Delta S_{total}|$, averaged over all feature vectors in the training set, versus the relative difference in classifier error between the NBC and GBC for each classifier. All correlation coefficients are statistically significant at the 95% confidence level.

Isn't Everything just a Building Block?

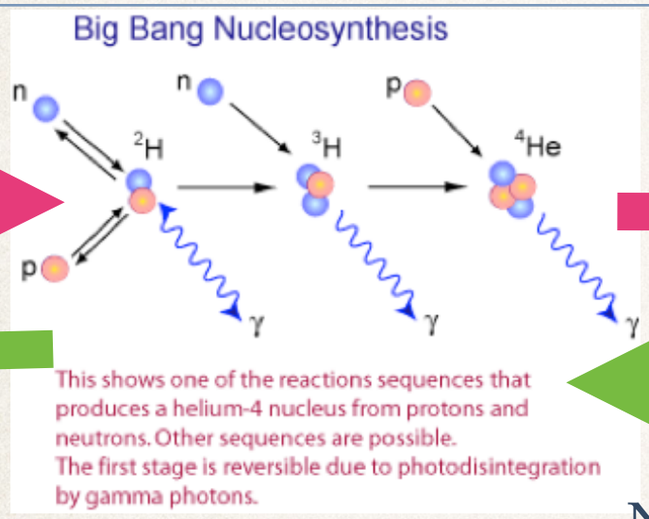




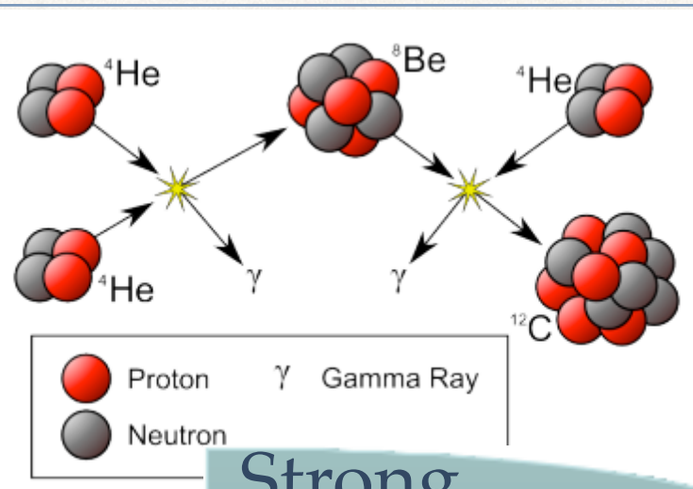




The Evolution of everything in the Universe seen as a Search process using Building Blocks

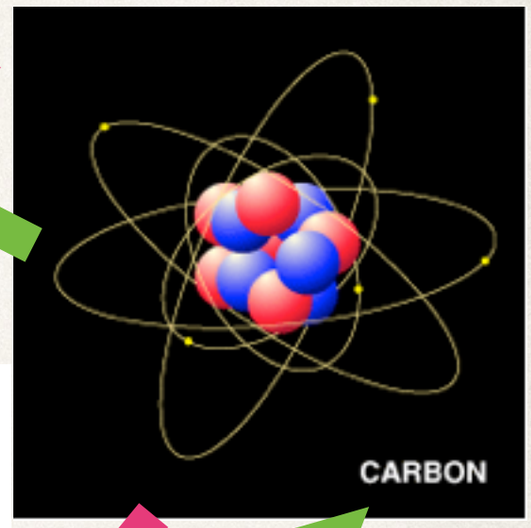


Particle physics
Less Complex



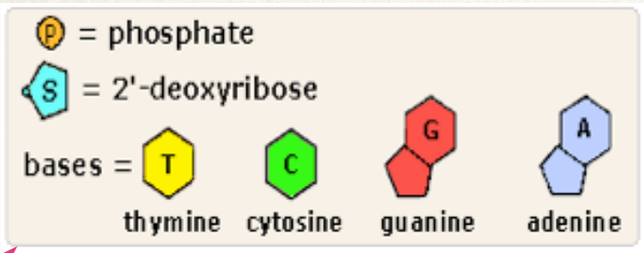
Nuclear physics

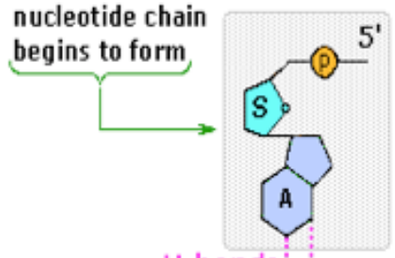
Atomic physics

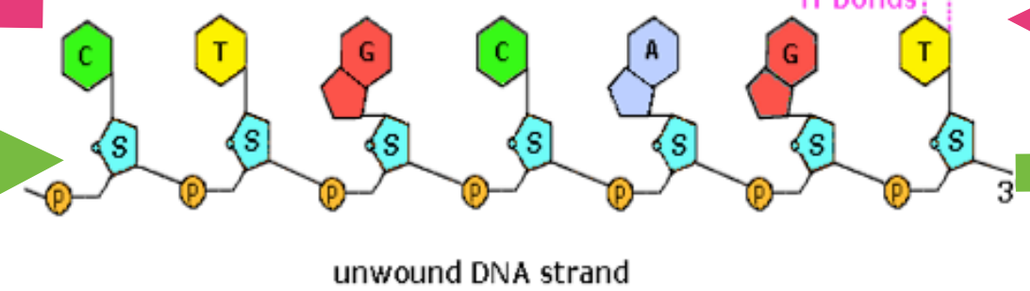


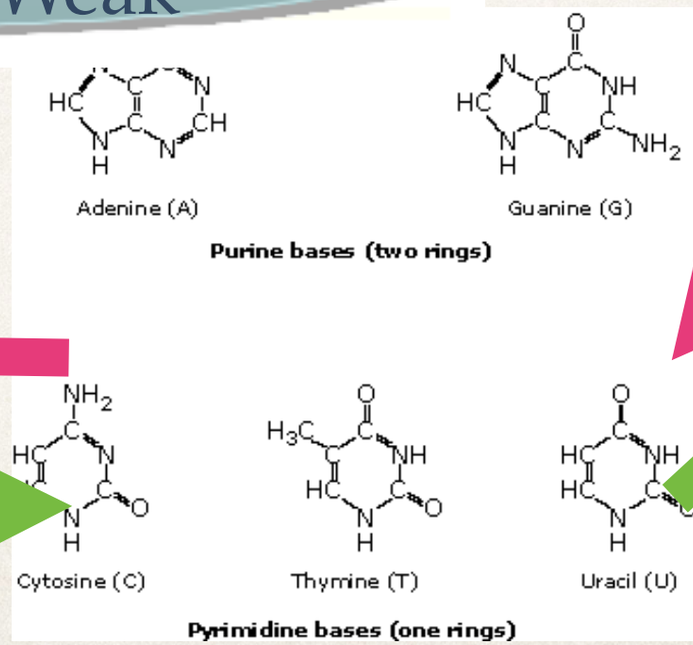
Strong interactions

Weak









Molecular Biology
More Complex

Organic chemistry

Why Building Blocks?

two watchmakers, Hora and Tempus, who make identical watches of 1000 parts. Hora's watches were composed of 10 sub-parts of 10 parts each, each of which in its turn was composed of 10 sub-parts. Tempus, on the other hand, makes watches without any subparts. Now, imagine that the watchmakers are interupted in their work. After an interuption Tempus has to go back to scratch. On the other hand, Hora has the advantage of only having to re-create the sub-assembly that was being made at the time, not the whole watch. Obviously, the sub-parts of Hora, the clever watchmaker, are building blocks, the construction of the watch using a building block hierarchy of depth two. If the probability of an interuption is p for every part constructed then the probability that Tempus will finish a watch is $(1-p)^{1000}$. For Hora, the probability that one of his subassemblies has to be reconstructed is $(1-p)^{10}$, as any subassembly, no matter where in the hierarchy, consists of 10 parts. The result of the two different construction strategies is that Tempus takes about 4,000 times more time to construct a watch than Hora.

Herbert A. Simon, The Architecture of Complexity, Proceedings of the American Philosophical Society, Vol. 106, No. 6., pp. 467-482 (1962).
Herbert A. Simon, The Sciences of the Artificial, 3rd edition, MIT Press, Boston, MA (1996).

Hora making He, Be and Li

$$
\begin{array}{ll}
n^0 \longrightarrow p^+ + e^- + \bar{\nu}_e & p^+ + n^0 \longrightarrow {}^2_1D + \gamma \\
{}^2_1D + p^+ \longrightarrow {}^3_2He + \gamma & {}^2_1D + {}^2_1D \longrightarrow {}^3_2He + n^0 \\
{}^2_1D + {}^2_1D \longrightarrow {}^3_1T + p^+ & {}^3_1T + {}^2_1D \longrightarrow {}^4_2He + n^0 \\
{}^3_1T + {}^4_2He \longrightarrow {}^7_3Li + \gamma & {}^3_2He + n^0 \longrightarrow {}^3_1T + p^+ \\
{}^3_2He + {}^2_1D \longrightarrow {}^4_2He + p^+ & {}^3_2He + {}^4_2He \longrightarrow {}^7_4Be + \gamma \\
{}^7_3Li + p^+ \longrightarrow {}^4_2He + {}^4_2He & {}^7_4Be + n^0 \longrightarrow {}^7_3Li + p^+
\end{array}
$$

Tempus making He

n+p+n+p —> He + gamma

Toy model of "nucleosynthesis" as search

Building Blocks and Search, Lozano, A., Mireles, V., Monsivais, D., Stephens, C.R., Alcala, S. and Cervantes, F., MICAI, 704-715, Springer-Verlag (2009).

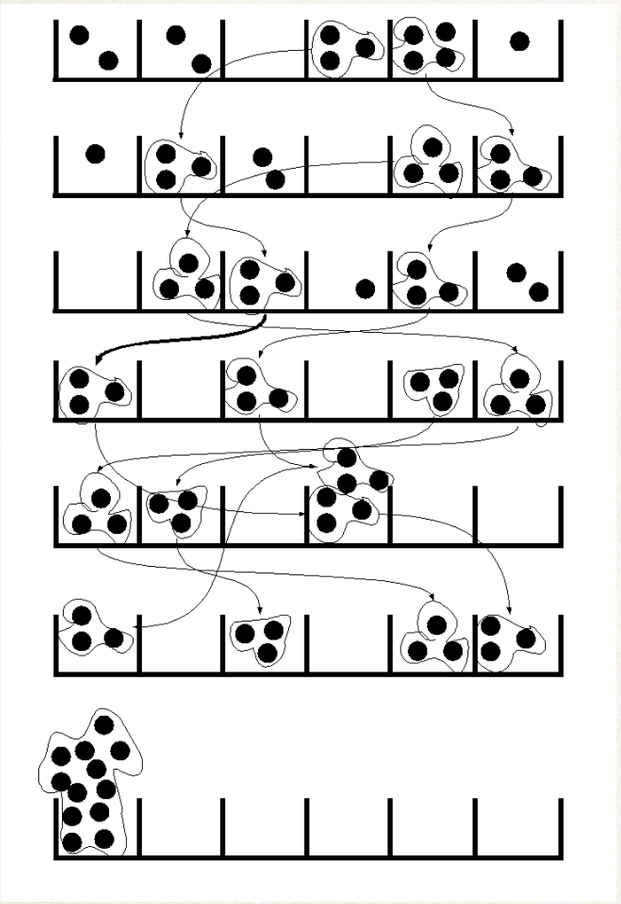


Fig. 1. A sample evolution of the presented model, with n = 12 particles and m = 6 lattice sites. Evolution is from top to bottom. The valid block sizes are {1, 3, 12}. Note how, when four particles fall on the same site, only a group of three is formed. Note also, how pairs of groups of three don't bind together, for 6 is not a valid block size.

- A set of n particles and m lattice sites.
- Construct states, where a certain, fixed number of particles are found at the same lattice site, this state being the objective of the search.
- What type of search algorithm is favoured?
- Search algorithms implementable as Markov processes that randomly permute building blocks between lattice points
- The question then becomes whether this binding of particles makes the algorithm more efficient and, if so, how does the nature of the building blocks -number and type - affect the efficiency of search?

$$V = \{v_1, v_2 \ldots, v_{k-1}, v_k\} \subset \{1 \ldots n\}$$

$$v_1 = 1,\ v_k = n \text{ and } v_i < v_j\ \forall i < j$$

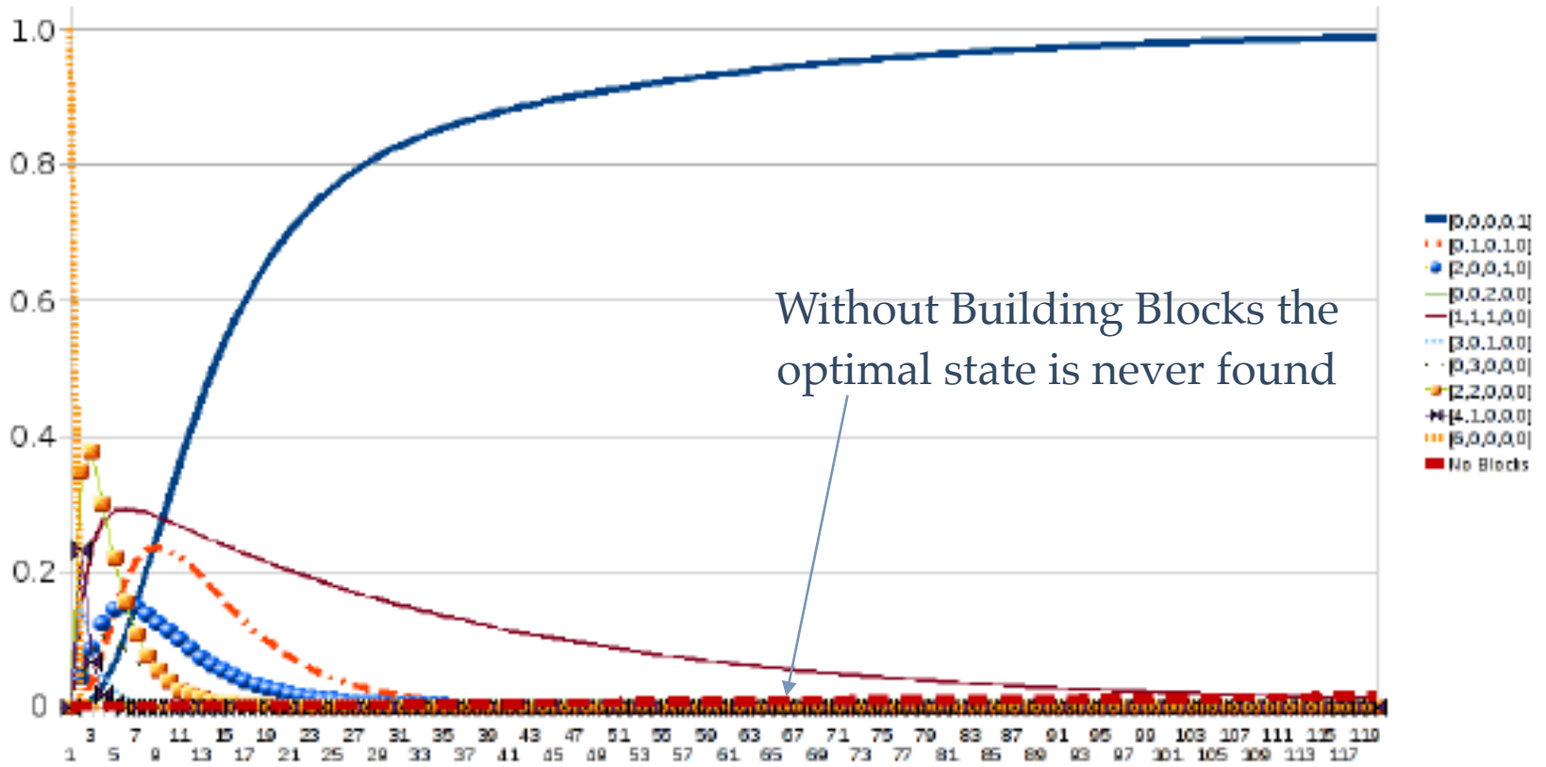
Can set up different building block types and number of levels

Toy model of "nucleosynthesis" as search

$$P(t+1) = MP(t)$$

Process is described by an upper-triangular Markov matrix. e.g.,

$$\begin{pmatrix}
1.000 & 0.167 & 0.028 & 0.167 & 0.028 & 0.005 & 0.028 & 0.005 & 0.001 & 0.000 \\
0.000 & 0.833 & 0.139 & 0.000 & 0.000 & 0.000 & 0.417 & 0.023 & 0.004 & 0.010 \\
0.000 & 0.000 & 0.833 & 0.000 & 0.000 & 0.000 & 0.000 & 0.139 & 0.000 & 0.042 \\
0.000 & 0.000 & 0.000 & 0.833 & 0.000 & 0.023 & 0.000 & 0.000 & 0.000 & 0.006 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.972 & 0.417 & 0.000 & 0.000 & 0.093 & 0.154 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.556 & 0.000 & 0.000 & 0.000 & 0.154 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.556 & 0.139 & 0.069 & 0.039 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.694 & 0.556 & 0.347 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.278 & 0.231 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.015
\end{pmatrix}$$



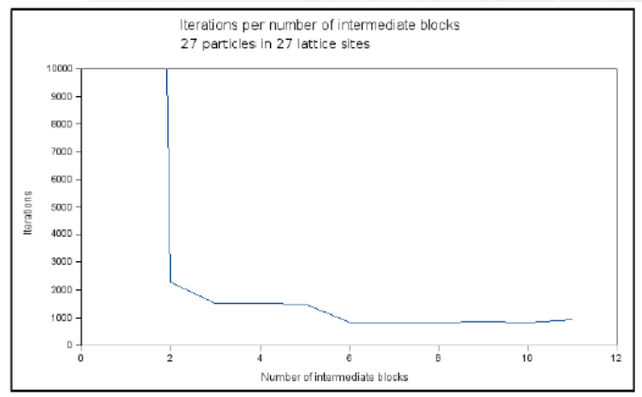
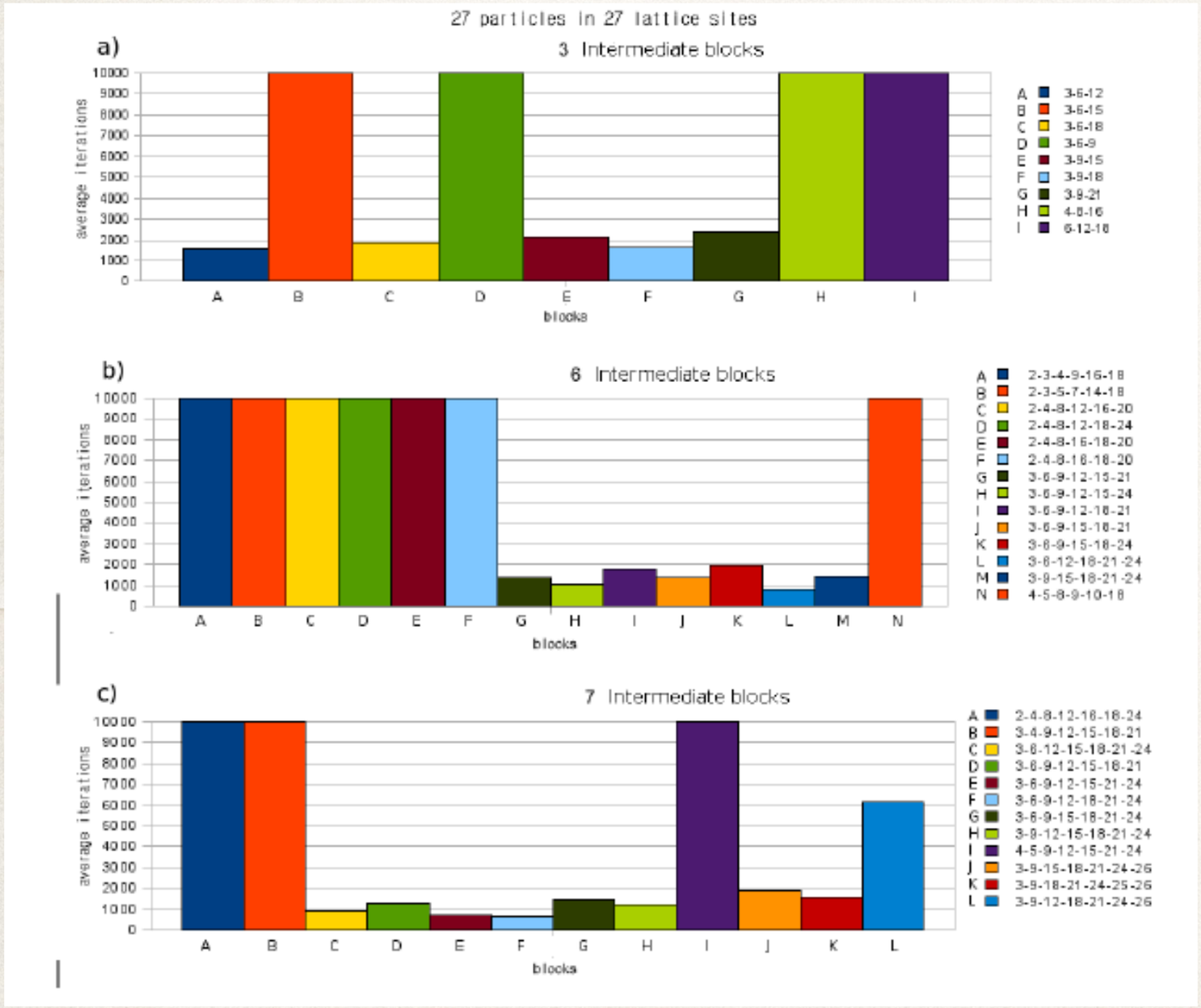


Fig. 3. *This figure shows the minimum mean number of iterations per number of valid blocks. For each number of blocks, 1000 iterations were made*

Fig. 2. *Evolution of each of the entries of the probability vector for $n = m = 6$ and $V = \{1, 2, 3, 4, 6\}$. The objective state is $[0, 0, 0, 0, 1]$. Also shown is the evolution of the probability of reaching the same objective state (all particles bound together) but using a system without building blocks. The x axis is time and the y axis is probability.*

The more Building Block levels the better, but asymptotes due to finiteness of supply of elemental blocks

Toy model of “nucleosynthesis” as search



Performance depends on both the number of building block levels and where they are located

Also depends on the availability of each block type. Missing blocks can make certain states inaccessible.

For an optimal state of 27 and 2 intermediate blocks, the optimal positions are 3 and 9 (geometric) as this equalises the difficulty of each step in evolution

Population Genetics as Search

Population of genotypes to be searched through

Search algorithms have selection, mutation and homologous recombination

$$P_I(t+1) = M_I{}^J((1-p_c)P'_J(t) + p_c\sum_m p_c(m)\lambda_J{}^{KL}(m)P'_K(t)P'_L(t))$$

$P'_I(t)$ Probability to select genotype I $\quad P'_I(t) = \frac{f(I)}{\bar{f}(t)}P_I(t)$

f(I) is the fitness of genotype I. The fitness landscape fixes the problem to be "solved"

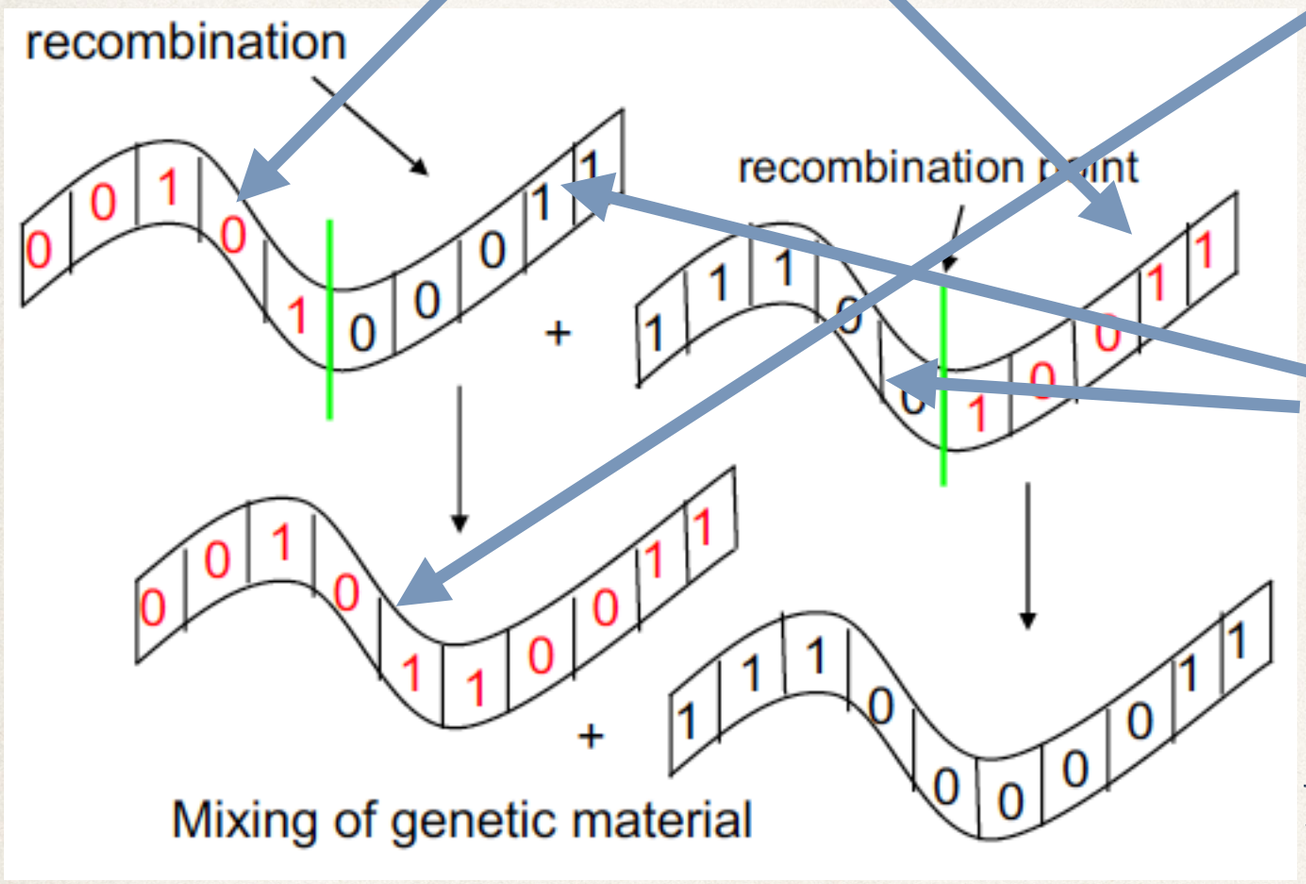
$M_I{}^J$	Probability to mutate genotype J to genotype I
p_c	Probability to implement recombination
$p_c(m)$	Probability that given recombination takes place it is implemented with mode m

These fix the search algorithm

Population Genetics as Search

Aesexual and Sexual Reproduction as Search Algorithms

These two **Building Blocks** make the genotype



We don't care what genetic material is here!

Dynamics is naturally described in terms of

$$\Delta_I(m) = P'_I - P'_{I_m} P'_{I_{\bar{m}}}$$

Selection-weighted Linkage Disequilibrium Coefficient

Just like the diagnostic for going beyond the NBA

Population Genetics as Search

What search algorithm is <u>good</u> on which fitness landscape (problem)?

What do we mean by good?

$$\mathcal{D}_{\bar{f}_{\mathrm{r+s}}}(t) = \overline{f}_{\mathrm{r+s}}(t+1) - \overline{f}_{\mathrm{s}}(t+1)$$

Is the average population fitness higher for one search algorithm versus another?

$$\mathcal{D}_{P_I}(t) = (P_I(t+1) - P_I(t))$$

Does one search algorithm produce more of a fit string than another?

Consider two genetic loci:

$$f_{x_1 x_2} = F^{(0)} + \sum_{i_1=1}^{2} F_{i_1}^{(1)} x_{i_1} + F_{12}^{(2)} x_1 x_2$$

$$\begin{aligned} f_{00} &= F^{(0)} = a, \\ f_{01} &= F^{(0)} + F_2^{(1)} = a + b_2, \\ f_{10} &= F^{(0)} + F_1^{(1)} = a + b_1, \\ f_{11} &= F^{(0)} + F_1^{(1)} + F_2^{(1)} + F_{12}^{(2)} = a + b_1 + b_2 + c. \end{aligned}$$

Describes all possible fitness landscapes (problems)

For an additive (modular) landscape $F_{12}^{(2)} = 0$. For a multiplicative landscape $F^{(0)} F_{12}^{(2)} = F_1^{(1)} F_2^{(1)}$. For a redundant (modular) landscape $F_{ij}^{(2)} = -F_i^{(1)} = -F_j^{(1)}$ which, as mentioned, can be understood in terms of a Boolean "OR", fitness being the same if either one or both alleles are optimal. For a NIAH landscape $F_1^{(1)} = F_2^{(1)} = 0$ which, in contrast to the redundant landscape, corresponds to a Boolean "AND" as fitness is only different if both alleles are optimal.

$b > -1$, $c > -2b$ and $c > -b$

Population Genetics as Search

Blue/red = recombination
Advantageous/disadvantageous

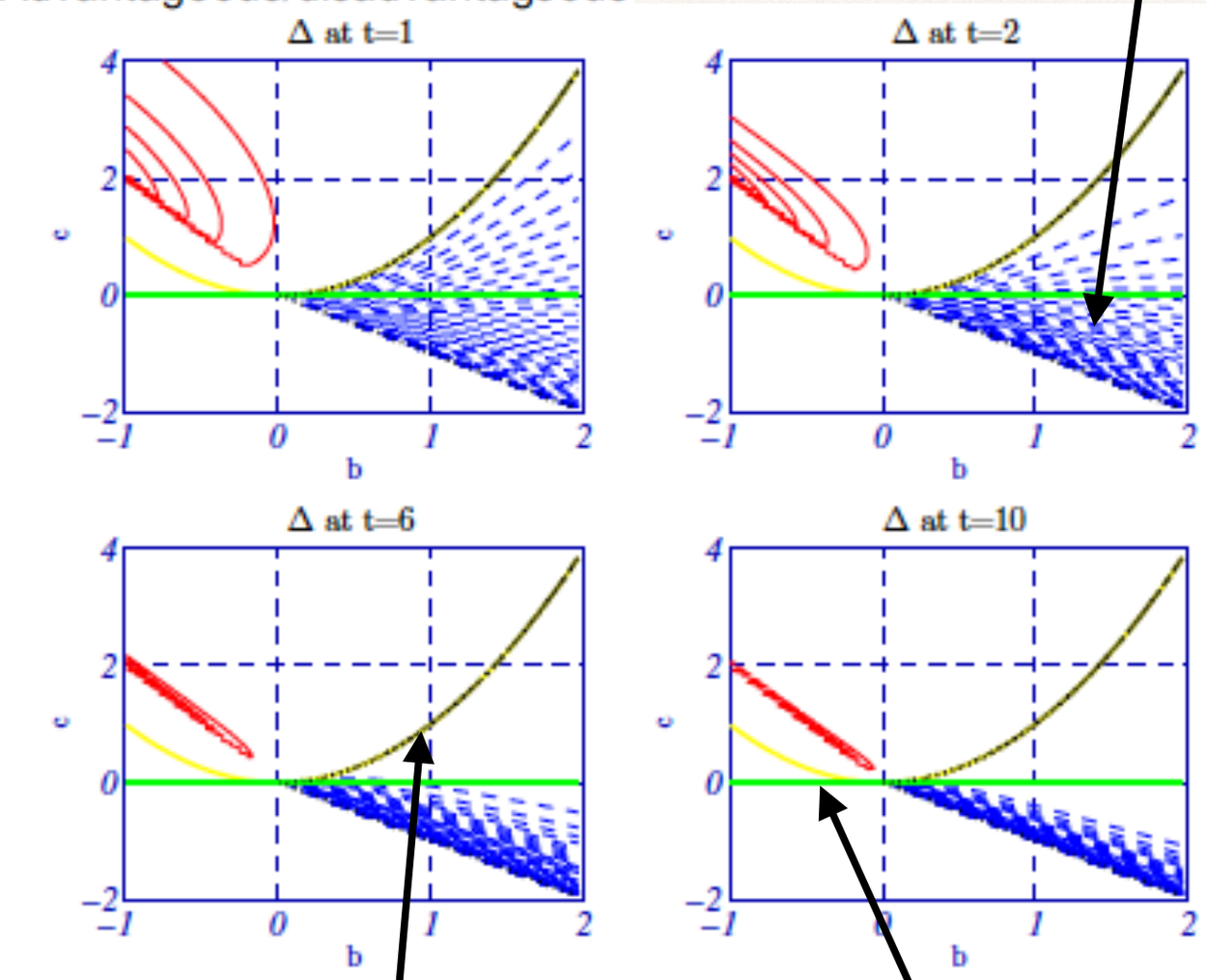


Fig. 8. Value of Δ at different generations for two-locus two-allele system as a function of fitness landscape, characterized by b and c. The initial population is $P_{ij}(0) = 0.25$. The $\Delta = 0$ plane has been marked to distinguish between conditions in which recombination is favorable ($\Delta < 0$) or not. The curve on the plane is $c = b^2$, the condition for a multiplicative landscape.

Sex better here

Blue/red = recombination
Disadvantageous/advantageous

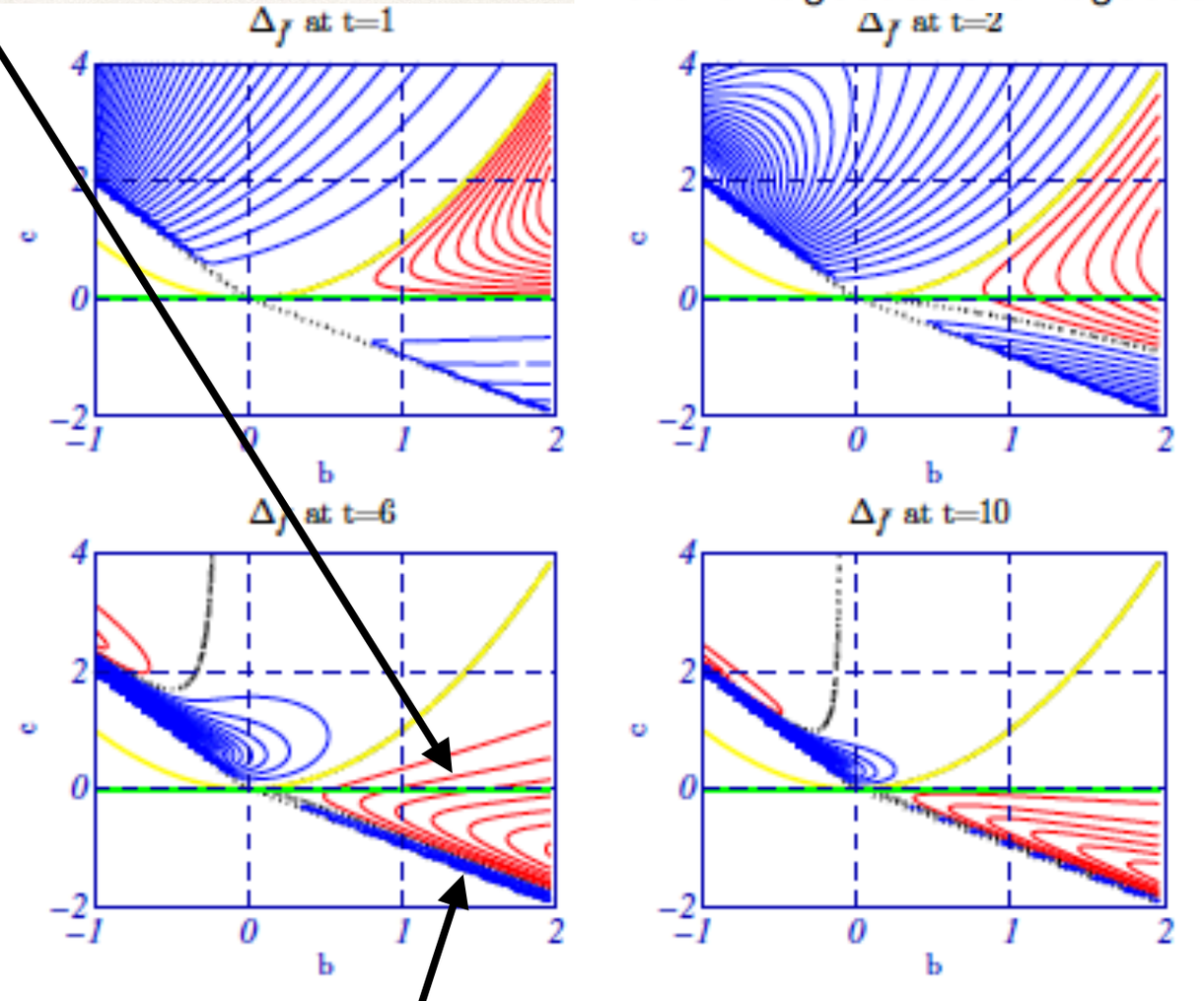


Fig. 9. Value of $\mathcal{D}_{f_{r+s}}$ at different generations for the two-locus two-allele system as a function of fitness landscape, characterized by b and c. The initial population is $P_{ij}(0) = 0.25$. The $\mathcal{D}_{f_{r+s}} =$ plane has been marked to distinguish between conditions in which recombination is favorab ($\mathcal{D}_{f_{r+s}} > 0$) or not.

Multiplicative landscapes

Additive landscapes

Maximum negative epistasis (redundancy)

Conclusions

- **Question**: What features of a given problem can tell us which search algorithm to use?
- Answer: As each search algorithm has a bias we need to see which features in the problem structure are inconsistent with that bias.
  - We showed how a set of correlation functions on the feature set yielded a lot of information about the underlying problem structure
- **Question**: Which search algorithms work best on which problems?
- Answer: Knowledge of problem structure allows one to predict which algorithm will offer best performance.
  - We showed how one could make an a priori differentiation between the NBA and its generalisations using knowledge of the problem structure.
- **Questions**: What problems are "special"? Is there anything special about "real world" (physics, biology,…) problems and/or search algorithms?
- Answer: Problems that have a hierarchical Building Block structure are special. Basically, all of physics and biology are of this type. They are associated with quasi-modular and redundant fitness landscapes. They are ubiquitous because they are the ONLY practical way to evolve complexity (from nucleosynthesis to genetics to social organisation)
- **Question**: Which search algorithms work best on these "special" problems?
- Answer: Recombinative search algorithms that combine building blocks, such as nucleosynthesis or sex.