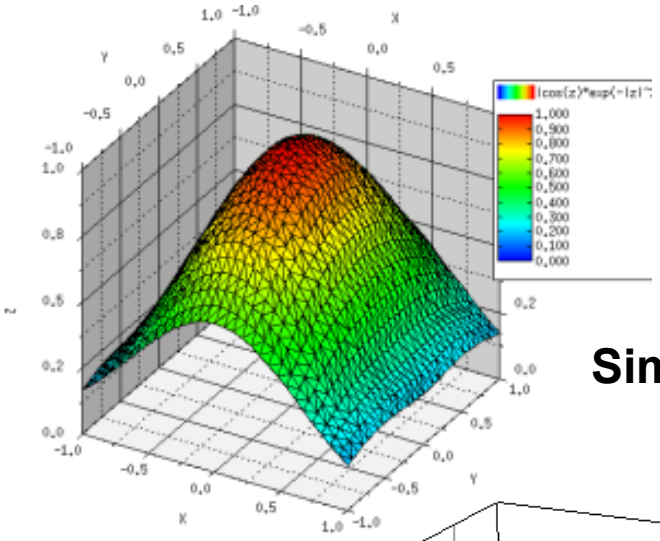
Optimization Algorithms

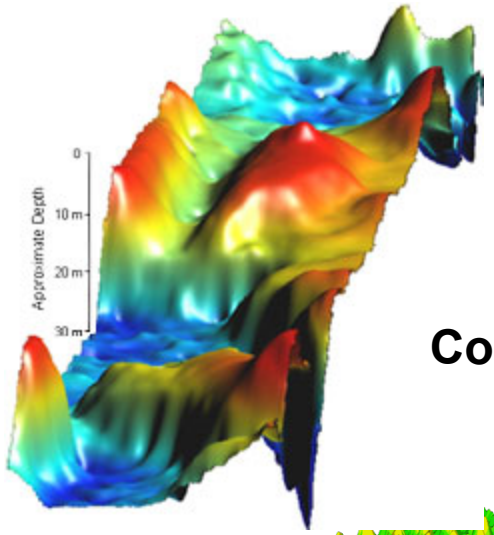
Kinds of optimization

- Univariate
  - Find the values of independent variables that produce a maximum (or minimum) value for a dependent variable
  - E.g. find the torsion angles for all the rotatable bonds in a molecule such that the energy is minimized
- Multivariate
  - Find the values of independent variables that produce a set of dependent variable values that are optimally minimized, maximized, or fit a pre-defined profile
  - E.g. find a 10% subset of a combinatorial library that contains compounds with logP, solubility and molecular weight that is closest to a particular desired profile

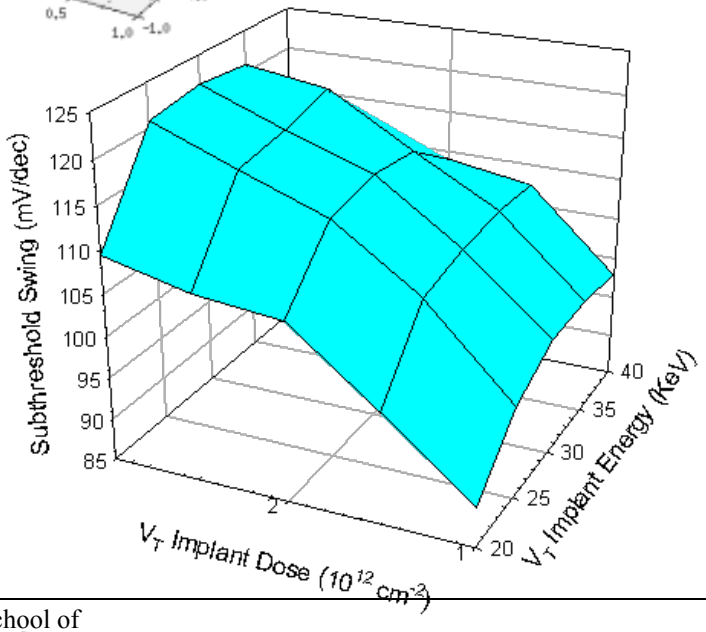
“Complexity” of optimization problems

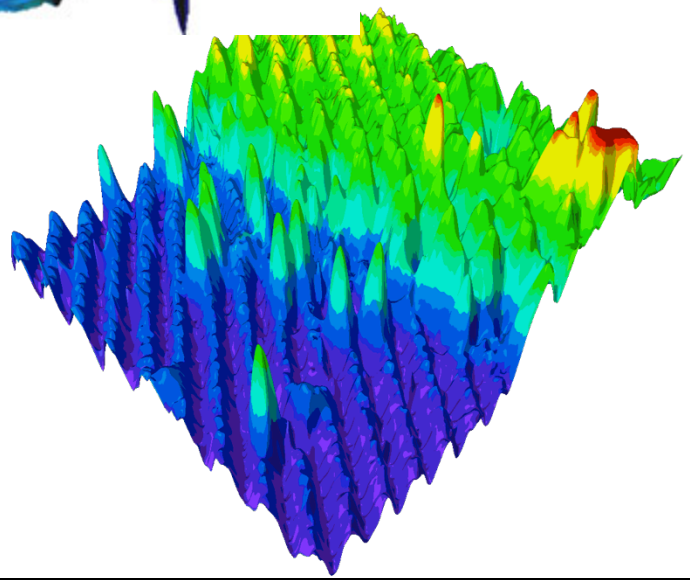


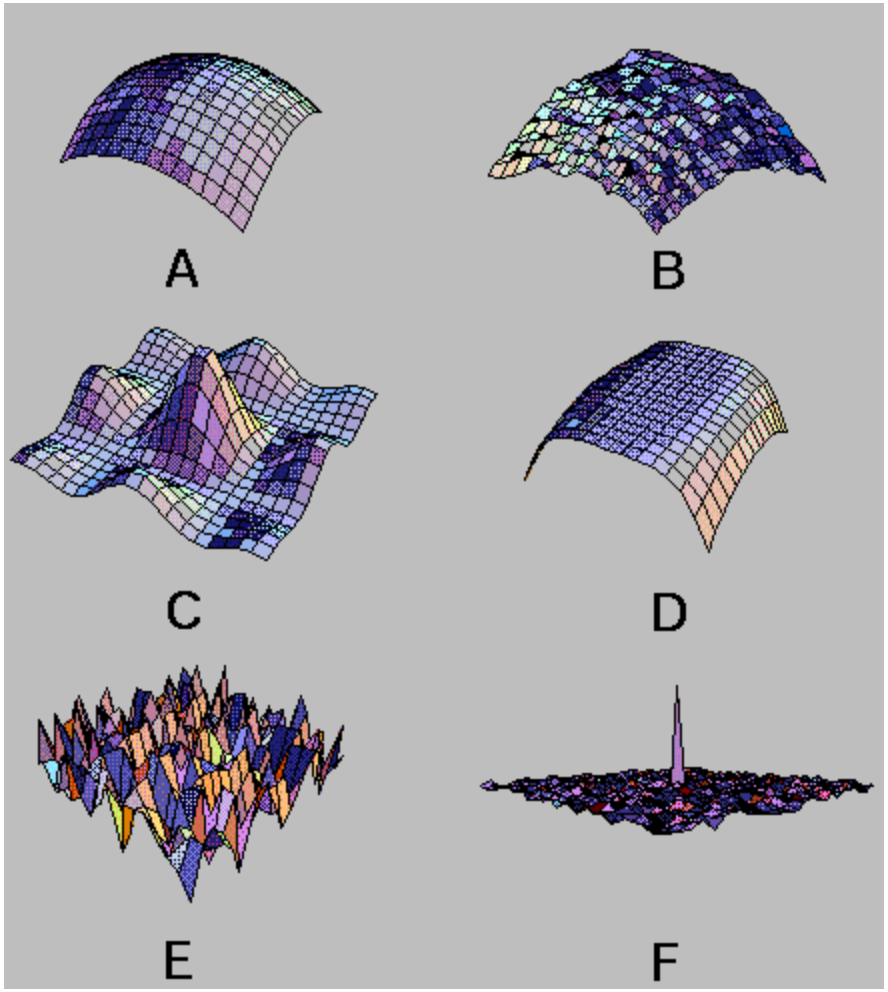
Simple



Complex







A: Unimodal functions.
B: Essentially unimodal functions.
C: Functions that have a small number of significant local optima.
D: Functions with significant null-space effects.
E: Functions with a huge number of significant local optima.
F: Functions whose global structure provides no useful information.

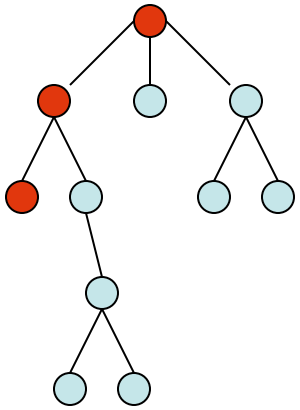
Taken from http://coool.mines.edu/report/node3.html

Optimization algorithms

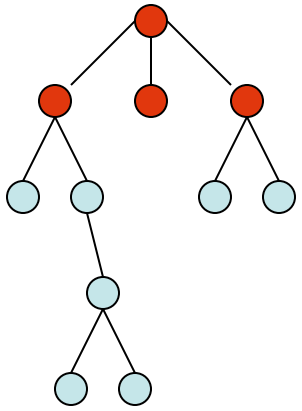
- Deterministic
  - Depth/Breadth/Best First Search
  - Hill climbing (steepest ascent/descent)
- Stochastic
  - Genetic algorithms and other Evolutionary Algorithms
  - Simulated annealing
  - Monte Carlo
  - Ant colony optimization
- For multivariate optimization
  - MOGAs (Multi-Objective Genetic Algorithms)

Depth / breadth / best first search

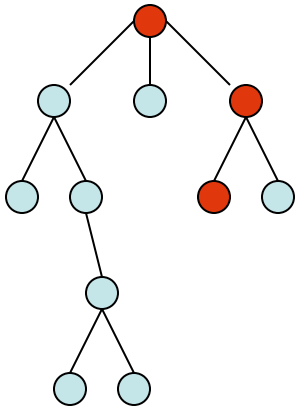
- Exhaustively test all possible states
- Tree / graph based – either go down branches first, or go across branches, or heuristically choose based on evaluation of nodes
- Requires bounded, finite search space
- Generally use recursive algorithm



DFS



BFS



BFS

Hill Climbing (steepest ascent/descent)

- Takes a single path to a local maximum / minimum

```
1. Pick an initial "state"
2. Evaluate all possible changes in state
3. If one has better value for dependent variable, make
that the current state and go to (2)
```

Genetic Algorithms

- Create a “population” of possible solutions, encoded as “chromosomes”
- Use “fitness function” to score solutions
- Good solutions are combined together (“crossover”) and altered (“mutation”) to provide new solutions
- The process repeats until the population “converges” on a solution
- http://en.wikipedia.org/wiki/Genetic_algorithm

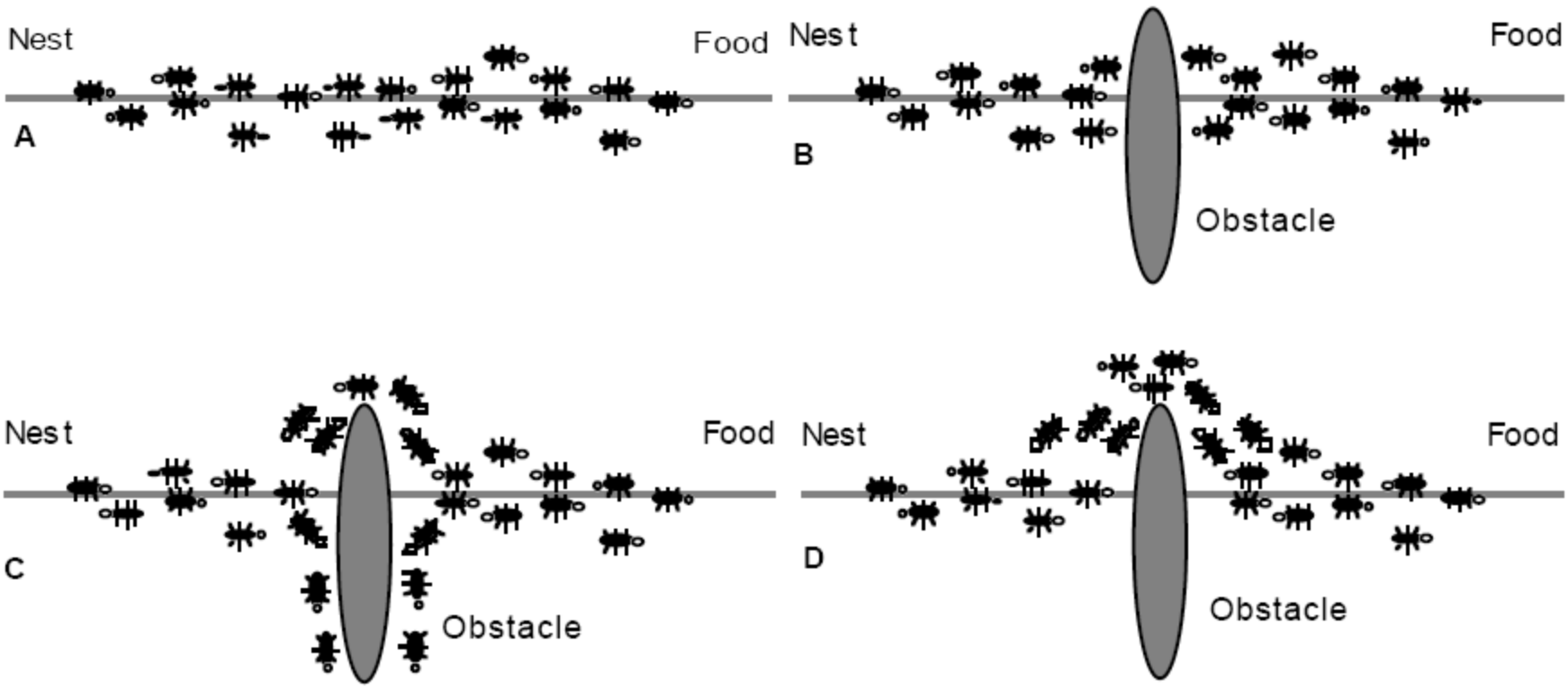
Simulated Annealing

- Based on manner in which liquids freeze or metals anneal: high temp = disordered, low temp = ordered, ground state at T=0
- Revolves around finding "T=0", i.e. "low energy state" or local minimum
- Chooses new states based on probabilities – higher chance of choosing a lower energy state, but may choose a higher energy state.
- http://en.wikipedia.org/wiki/Simulated_annealing

Monte Carlo Method

- Multiple “random walks” through landscape
- Each random walk makes entirely random changes of state
- Tends towards areas of landscape with better values for depenedent variable
- http://en.wikipedia.org/wiki/Monte_Carlo_method

Ant Colony Optimization

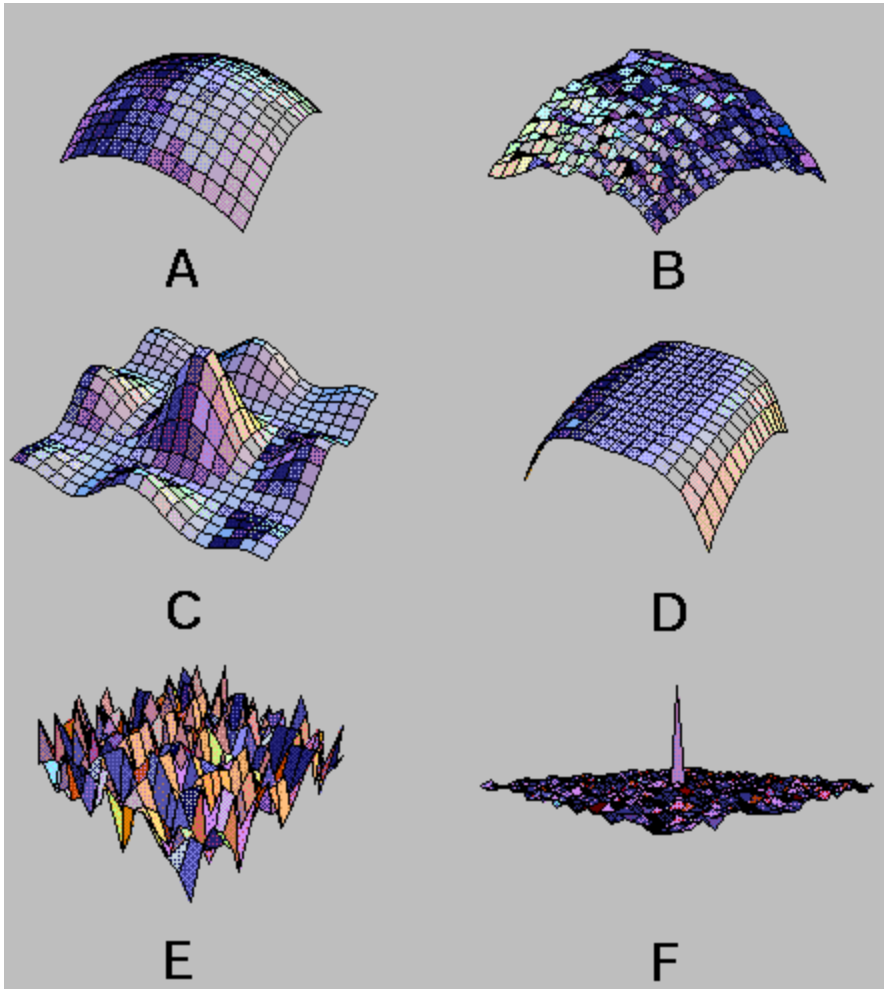


- http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html
- http://uk.geocities.com/markcsinclair/aco.html

Ant Colony Optimization

These are three ideas from natural ant behavior that we have transferred to our artificial ant colony: *(i)* the preference for paths with a high pheromone level, *(ii)* the higher rate of growth of the amount of pheromone on shorter paths, and *(iii)* the trail mediated communication among ants. Artificial ants were also given a few capabilities which do not have a natural counterpart, but which have been observed to be well suited to the TSP application: artificial ants can determine how far away cities are, and they are endowed with a working memory *Mk* used to memorize cities already visited (the working memory is emptied at the beginning of each new tour, and is updated after each time step by adding the new visited city).

Which methods would work with which landscapes?



Hill Climbing
X-first search
Genetic Algorithm
Simulated Annealing
Monte Carlo
Ant Colony Optimization

How GAs Work

A way is found of encoding possible solutions into a bitstring (chromosome), and of specifying the 'goodness' of a chromosome (fitness function)

1. *Initialize a population of chromosomes*
2. *Evaluate the fitness of each chromosome*
3. *Create new chromosomes from the current population*
4. *Delete population members to make room for new ones*
5. *Evaluate the new chromosomes and put them in population*
6. *If we want to keep going, go back to step 3*

Example: http://homepage.sunrise.ch/homepage/pglaus/gentore.htm

Genetic Algorithm Example

- For our purpose, we can encode rotation and translation of a molecule, and bond torsion angles in a chromosome, e.g:

$$T_X \quad T_Y \quad T_Z \quad R_X \quad R_Y \quad R_Z \quad \tau_1 \quad \tau_2$$

where we have 3 translation values (T), 3 rotation values (R) and as many torsion angles (τ) as the molecule has rotatable bonds

Initial Population

- Initially, our population will be initialized with random values, e.g.

	T_X	T_Y	T_Z	R_X^o	R_Y^o	R_Z^o	τ_1^o	τ_2^o
C1	-3.2	-1.6	4.5	130	126	228	131	114
C2	2.8	1.3	-4.6	97	231	149	126	144
C3	-8.7	2.9	3.1	143	261	12	83	29
C4	-2.2	-2.9	-3.6	27	280	141	312	216
C5	5.8	4.1	4.9	19	25	26	341	18
C6	0.3	-2.7	5.6	14	81	27	155	75
C7	4.4	-0.3	-0.2	12	46	22	26	98

Fitness Function

- Used to score chromosomes to determine "goodness"
- For our purposes, we're concerned with how well the molecule in a particular orientation binds to the protein
- So a fitness function for a docking GA might be a combination of the following elements:

  - Energy (binding, potential)
  - Number and strength of hydrogen bonds formed
  - Hydrophobic effects
  - Electrostatic effects

Fitness Function

- To score a chromosome, the GA will place the molecule inside the protein using the given translation, rotation and torsion parameters, and the fitness function will calculate the score based on an analysis of the joint 3D structure

Fitness function scoring of population

- Initially, our population will be initialized with random values, e.g.

	T_X	T_Y	T_Z	R_X^o	R_Y^o	R_Z^o	τ_1^o	τ_2^o	**Score**
C1	-3.2	-1.6	4.5	130	126	228	131	114	**0.42**
C2	2.8	1.3	-4.6	97	231	149	126	144	**0.95**
C3	-8.7	2.9	3.1	143	261	12	83	29	**0.87**
C4	-2.2	-2.9	-3.6	27	280	141	312	216	**0.04**
C5	5.8	4.1	4.9	19	25	26	341	18	**0.32**
C6	0.3	-2.7	5.6	14	81	27	155	75	**0.78**
C7	4.4	-0.3	-0.2	12	46	22	26	98	**0.61**

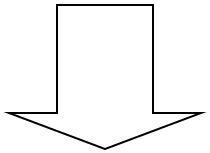
Create new population members

- Initially, our population will be initialized with random values, e.g.

	T_X	T_Y	T_Z	$R_X°$	$R_Y°$	$R_Z°$	$\tau_1°$	$\tau_2°$	**Score**
C1	-3.2	-1.6	4.5	130	126	228	131	114	**0.42**
C2	2.8	1.3	-4.6	97	231	149	126	144	**0.65**
C3	-8.7	2.9	3.1	143	261	12	83	29	**0.77**
C4	-2.2	-2.9	-3.6	27	280	141	312	216	**0.04**
C5	5.8	4.1	4.9	19	25	26	341	18	**0.32**
C6	0.3	-2.7	5.6	14	81	27	155	75	**0.78**
C7	4.4	-0.3	-0.2	12	46	22	26	98	**0.61**

Crossover

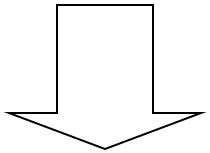
C2	2.8	1.3	-4.6	97	231	149	126	144	**0.65**
C3	-8.7	2.9	3.1	143	261	12	83	29	**0.77**



C8	-8.7	1.3	-4.6	97	261	12	83	144	

Mutation

C2	2.8	0.3	-4.6	107	221	149	126	134	**0.65**



C9	2.8	0.4	-4.6	105	226	149	126	122	

Score new chromosomes

	T_X	T_Y	T_Z	R_X°	R_Y°	R_Z°	τ_1°	τ_2°	**Score**
C1	-3.2	-1.6	4.5	130	126	228	131	114	**0.42**
C2	2.8	1.3	-4.6	97	231	149	126	144	**0.65**
C3	-8.7	2.9	3.1	143	261	12	83	29	**0.77**
C4	-2.2	-2.9	-3.6	27	280	141	312	216	**0.04**
C5	5.8	4.1	4.9	19	25	26	341	18	**0.32**
C6	0.3	-2.7	5.6	14	81	27	155	75	**0.78**
C7	4.4	-0.3	-0.2	12	46	22	26	98	**0.61**
C8	-8.7	1.3	-4.6	97	261	12	83	144	**0.83**
C9	2.8	0.4	-4.6	105	226	149	126	122	**0.56**

Delete poor chromosomes

	T_X	T_Y	T_Z	R_X°	R_Y°	R_Z°	τ_1°	τ_2°	**Score**
C1	-3.2	-1.6	4.5	130	126	228	131	114	**0.42**
C2	2.8	1.3	-4.6	97	231	149	126	144	**0.65**
C3	-8.7	2.9	3.1	143	261	12	83	29	**0.77**
~~*C4*~~	~~-2.2~~	~~-2.9~~	~~-3.6~~	~~27~~	~~280~~	~~141~~	~~312~~	~~216~~	~~**0.04**~~
~~*C5*~~	~~5.8~~	~~4.1~~	~~4.9~~	~~19~~	~~25~~	~~26~~	~~341~~	~~18~~	~~**0.32**~~
C6	0.3	-2.7	5.6	14	81	27	155	75	**0.78**
C7	4.4	-0.3	-0.2	12	46	22	26	98	**0.61**
C8	-8.7	1.3	-4.6	97	261	12	83	144	**0.83**
C9	2.8	0.4	-4.6	105	226	149	126	122	**0.56**