

Re: Haldane's Dilemma – clarifications – and Felsenstein [LONG]

Source: <http://sci.tech–archive.net/Archive/sci.bio.evolution/2006–06/msg00306.html>

- *From:* "Perplexed in Peoria" <jimmenegay@xxxxxxxxxxxxxxx>
 - *Date:* Sun, 25 Jun 2006 17:16:27 –0400 (EDT)
-

"Malcolm" <regnizar@xxxxxxxxxxxxxxx> wrote in message
[news:e7jp1h\\$2g2m\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:e7jp1h$2g2m$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Here we go.

You are putting a lot of work into this. I hope you won't take my negative comments too, ummh..., negatively.

[snip]

One interesting thing to note is that, in this model, when the number of genes is large – I originally set it to 30000 to represent the number thought to exist in the human genome, then the number of mutations that make it into the population is approximately $2 * \text{SELECTIONCOEFFICIENT}$. When I tried to speed the program up by a factor of 300 by reducing the number of genes to 100, this relationship disappeared.

Maybe this is worth pursuing.

This sounds like a consequence of the theory of 'nearly neutral' selection. With lots of loci, the count of the number of improvements at any locus rarely rises above +1 or falls below –1. With a low selection coefficient, and a low population size, most loci are nearly neutral. But when the number of loci falls to a small fraction of the number of generations, every locus eventually accumulates a significant count of improvements, and most loci are no longer nearly neutral.

The number 250000 represents the number of generations since the human / chimp split, or as ReMine would say, the alleged human/chimp split.

If your goal here is to convince ReMine he is wrong about humans and chimps, I suspect you are wasting your time and your computer's time. For one thing,

Re: Haldane's Dilemma – clarifications – and Felsenstein [LONG]

you would have to change your model (drastically) so that it more closely matches hominid life histories. But if the goal is to understand whether the rate of substitution is really limited in a model with fertility selection, then I think that it can be done using a much smaller number of generations.

However this is only a model. it can be made better, for instance at the moment males and females are absolutely symmetrical, whilst we know that in human populations males have more variable fertility. So let's hope Walter ReMine can, as Joe Felsenstein has done, suggest improvements to show the effect he wants to show.

Don't hold your breath. Walter declined to respond to your model before, and will probably do so again.

[snip]

```
/*
choose a parent
Params: candidates – list of potential parents
N – number in list
Returns: the selected parent
We take two at random, and return the one with the highest fitness,
or the second to be examined if equal.
*/
int choose(CREATURE *candidates, int N)
{
double target;
int top;
int bottom;
int mid;

target = uniform() * candidates[N-1].cumfitness;

bottom = 0;
top = N-1;
while(bottom != top && bottom != top-1)
{
mid = (top + bottom)/2;
if(candidates[mid].cumfitness < target)
bottom = mid;
else
top = mid;
}
if(candidates[mid].cumfitness < target)
mid++;

if(mid > 0 && candidates[mid-1].cumfitness > target)
mid--;
```

```
return mid;  
}
```

I had a devil of a time figuring this out. It didn't help that the block comment on this function is completely out of date with what the function does now. When I finally did figure out what you were doing, I had to admit that I found it rather clever. But it is not the way I would have done it. Instead of keeping a cumulative (running total) fitness for each individual in the population, I would instead keep a single MAX fitness for the population. I would randomly select a candidate parent as you used to do. The candidate is tested with a probability of 'passing' being the ratio of his/her fitness to the MAX fitness. If the candidate 'fails', try another candidate until one passes. My approach is more efficient, but not as clever as yours.

[snip]

```
for(i=0;i<NGENES;i++)  
{  
fitness += child->matchchromosome[i];  
fitness += child->patchromosome[i];  
}  
  
child->fitness = pow(1 + SELECTIONCOEFFICIENT, fitness);
```

This code snippet, like the out-of-date comments, illustrates how code readability deteriorates over time. You are using the symbol 'fitness' here with two very different meanings. One is an int count of favorable mutations. The other is a double with a meaning more in the Fisher-Wright tradition. Confusing to the reader.

[snip remainder]

Have you tried multiple runs, varying the mutation rate as I suggested?

.