Battle Of Sexes

Implementing an evolutionary system in java

Vincenzo Mensitieri, Emanuele Scaccia, Stefano Zappa, Giordano Tambara

Sapienza Università di Roma

1. Introduction

This project is inspired by a book by Richard Dawkins named **"A selfish gene"**.

The aim of this project is to simulate an evolutionary system starting with an initial population to which the evolutionary rules will be applied, until the population reaches a state of stability. The population evolves according to precise evolution rules that will be furtherly explained later.

The population is divided into 4 types:

- **Men**
  - *Faithful:* prefers long courtship and they take care of their children;
  - *Philanderer:* doesn't want a long courtship, if they are not immediately accepted they move on without taking care of their children;
- **Woman**
  - *Fast:* doesn't mind copulating without a courship, independently on the type of man;
  - *Coy:* needs a long time courtship.

The relationship between men and women is based on a process of matching which is more favorable for them, based on the concept of **payoff**.

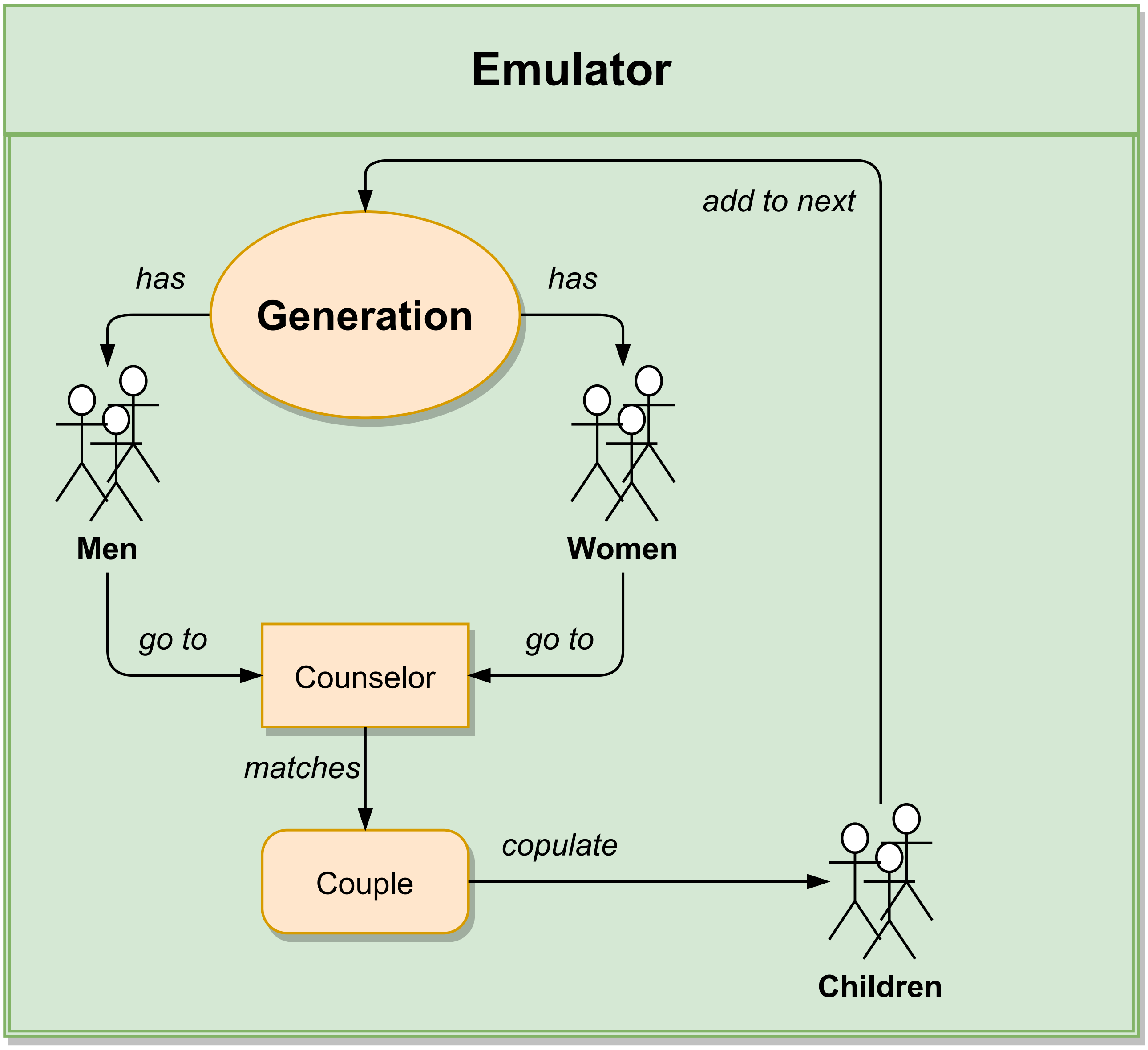
Payoffs are calculated by specific equations taking three parameters:

- **a,** the benefit of having a baby;
- **b,** the cost of having a baby;
- **c,** the cost of courtship.

	F	**P**
C	$(a - \frac{b}{2} - c,\ a - \frac{b}{2} - c)$	$(0, 0)$
S	$(a - \frac{b}{2},\ a - \frac{b}{2})$	$(a - b, a)$

A state of stability is reached when a generation has very similar ratios of types to the previous generations.

2. Adopted Model



2.1. Setting

The setting of the project has been chosen to be built around the concept of a **Government** that manages all the matchings. This Government is supported by institutions that supervise the phases the individuals have to pass through.

An important figure is the **Counselor**: he is meant to oversee the matching procedure, aiming to assign the most suitable option for everyone. A Counselor may serve just for one generation, after he has finished matching the people he will be retiring.

2.2. Matching

In the model adopted in this project, courtship was decided to be unilateral: men decide if they want to propose to a woman, and women decide whether to accept or not the proposal.

It is important to highlight that matching is determined by both sexes' preference.

In fact, our model is inspired by the **Stable Matching Problem** and, consequently, shares core features with the Gale-Shapley Algorithm: a parametrized number of men (default number was chosen to be 2) of random types interact with the same number of randomly picked women.

Every man expresses a predilection towards its possible partner: with most triplets of parameters, there will be a union that is better for him than the other one based on the payoff he will get for being with that type of woman. The bigger payoff, the better.

The same exact modality happens for the other sex; nevertheless, the choice is limited by the random pick of the individuals involved; in fact, during a single match, we can end up with every person of the same sex being of the same type. In that case, there is just one "obliged" option to select.

The main difference with the original Gale-Shapley Algorithm is that, during a cycle, we don't want to necessarily pair every "participant" immediately. Instead, we couple only the individuals that prefer each other; if not coupled, the person will be sent again to the *Counselor* and re-picked for another cycle.

2.3. Exceptions

Obviously, coherently to what was precedently stated, Philanderer men and Coy women cannot interbreed in any case; this is taken in consideration in the code: in fact they will never form a couple.

Furthermore, in theory, the Philanderer men shouldn't form a couple of any kind; according to Dawkins' book, they should leave their partner after copulating. Nevertheless, given that they can pair only with Fast women, for the sake of the project, there isn't any practical difference in mating with different women or being in a monogamous relationship.

Everyone is given the same opportunity to have a defined number of children, so, doing it with the same Fast woman or with different ones will not change the result.

2.4. Children

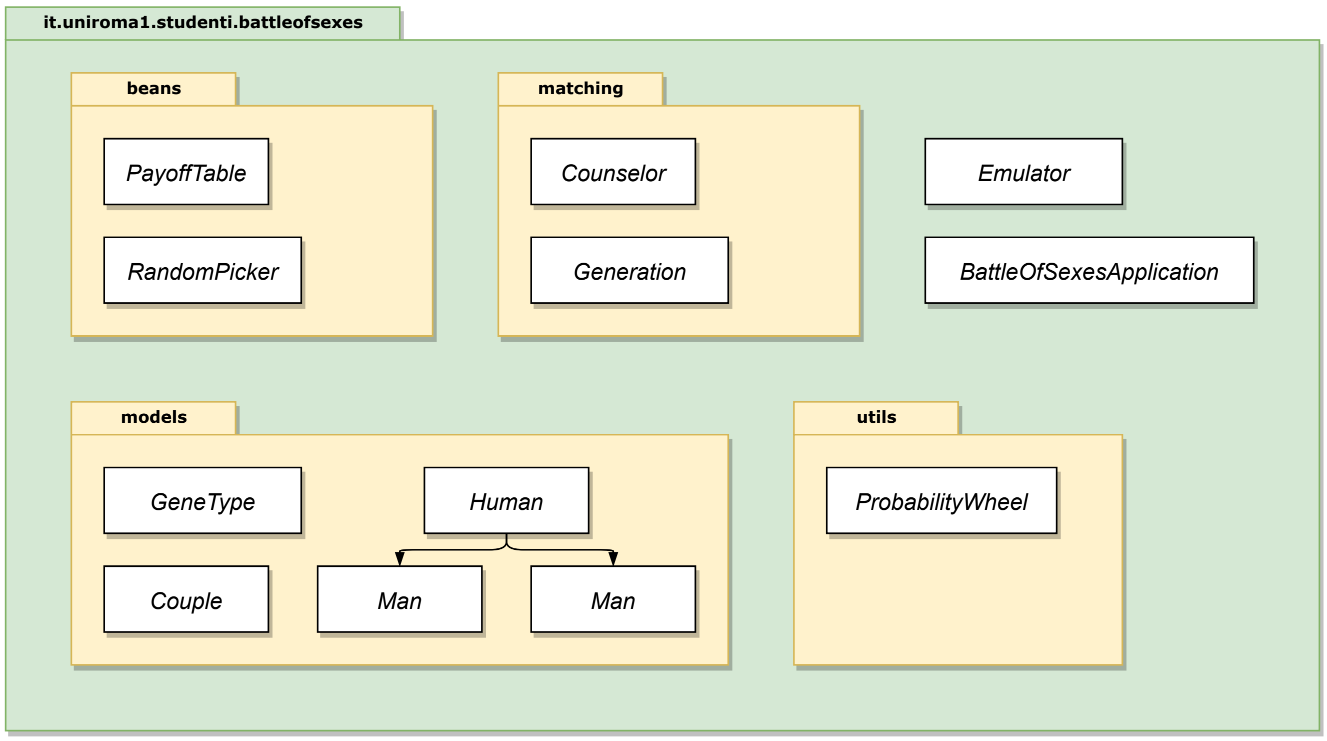
On the other hand, in case a couple is formed, it will procreate and pass their children to the next generation.

The number of children varies from 1 to 4, with every option having a parameterizable percentage to happen (Default values have been chosen to be 10% probability to have just one child, 20% for two, 50% for three and 20% for four).

These values have been chosen in order to have a slight increase in the population when we pass to the next generation (the average is 2.8 children).

3. Software Structure

In the main package we have 4 subpackages (each with its classes) and 2 main classes.



3.1. BattleOfSexes

3.1.1. BattleOfSexesApplication:

This is the first code to be run, it mostly contains the parameterizable properties and variables. Automatically, **Spring** executes the methods marked with `@Autowired`, setting the parameters.

3.1.2. Emulator:

Emulator is the core of the project: it starts from a random population (always based on given probabilities) and computes the next generations.

The method `isStable()` checks if the current and the previous populations have similar ratios among the various types: in that case, we say that the population has reached a state of **evolutionary stability**.

For every generation, it prints the total population and the ratios (in percentage) of the types, until it reaches evolutionary stability or for 100 generations.

3.2. Beans

This package contains components used to collect data.

3.2.1. PayoffTable:

This class is used to gather the data from the Payoff Table, namely to compute the payoffs of every type of individual in every single situation.

We initially define 3 `float` parameters: **a**, **b**, **c**, these will be used to calculate the payoffs for each pair of types based on the respective formula. Thus, 8 methods have been implemented to return every single value we need, two for each pair (4 pairs): one man to woman, one to woman to man.

Plus, there is a method `get()` that takes as parameters two ordered types and returns the respective payoff value.

3.2.2. RandomPicker

This class contains the methods that are needed to randomize different parts of the simulation.

We set the attributes `sons`, maleGenes and femaleGenes as instances of `ProbabilityWheel`; we create 3 methods (`setSons(...)`, `setMaleGenes(...)`, `setFemaleGenes(...)`) that Spring uses to set the data based on the given probabilties.

The other 3 methods `sons()`, `maleGenes()`, `femaleGenes()` return a random value of the regarding type based on the given probabilities.

3.3. Matching

3.3.1. Counselor

Counselor, as we said, extends `Thread`, and it is run asynchronously.

When used in the `compute()` method of Generation, a number (by default 4) of them are created and run simultaneously and stopped when they finish or after a timeout of 3 seconds.

Until we have enough people, we run an algorithm similar to the Gale-Shapley for 3 random women and 3 random men; the only difference is that the only couples formed are the ones that match the partners' top preferences.

3.3.2. Generation

Generation is a synchronous object that takes as arguments a list of Man and a list of Women and puts them in two separated queues.

The purpose of this object is to generate, through the method `compute()`, the next generation of children based, obviously, on the current generation and its characteristics.

The method `compute()` uses a specified number of Counselors, waits for them to finish or stops them after a timeout, after which we'll have the children which will be the population of the next generation.

Other methods contained in this class are `getTypeRatios()`, that returns the various percentages of the different types in the new generation under a `Map`, `getTotalChildren()` that returns the total number of them and `toString()` which returns verbose information about the generation.

Lastly, we have `addCouple(...)`, through which the Counselors can register the children a couple to take part in the next generation.

3.4. Models

This package contains the classes determining the individuals' attributes and their behaviour.

3.4.1. GeneType

Genetype is a functional interface that only has one method: `getCode()`.

Being an interface, it lacks the actual implementation which will be written when called in specific situations.

3.4.2. Human

This is an abstract class whose constructor takes the GeneType type object as an argument.

From this class we will derive the classes Man and Woman.

3.4.3. Man

This class extends `Human`, so it gets the attributes from its superclass; we define the `getType()` method which returns the type associated to the Man object.

Here has been implemented the `topPreferences(...)` method that, based on the payoff that Women would get by being with a specific type of man, ranks the possible partners and returns the best option for the woman taken in consideration.

Then we define the instance enum Type, which implements Genetype; here we define the two types of genes a man can have: `Faithful(F)`, `Philanderer(P)`.

3.4.4. Woman

This class is basically the same thing as Man but it obviously contains the `topPreferences(...)` method for the Men, which returns the best type of woman for the man taken in consideration based on the value of the payoffs, and it defines the genes regarding women: `Coy(C)`, `Fast(S)`.

3.4.5. Couple

In this class we define a Man and a Woman from their classes and a set of Humans, which are the children.

A couple takes as argument the father and the mother and contains an initially empty set of children; with the method procreate, we generate the offspring from the partners taken as argument: the number is random and defined by the `RandomPicker` class which we talked before, and the sex has a 50/50 chance.

Sons will inherit the father's type, Daughters the mother's one.

3.5. Utils

3.5.1. ProbabilityWheel

This is a utility class, used to pick random items with determined probabilities.

It takes a generic parameter `<T>`, which represents the type of object that will be picked. We use one map to store the values of the probabilities, in percentage, regarding the items, called "originals" and one `LinkedHashMap` to store the decimal probability value.

It has 3 methods. The first and most important one is `spin()`; this method "spins the wheel" returning a random object based on the given probabilities. Then we have the methods `getProbabilities()` that returns originals, and `getRanges()` that returns ranges.

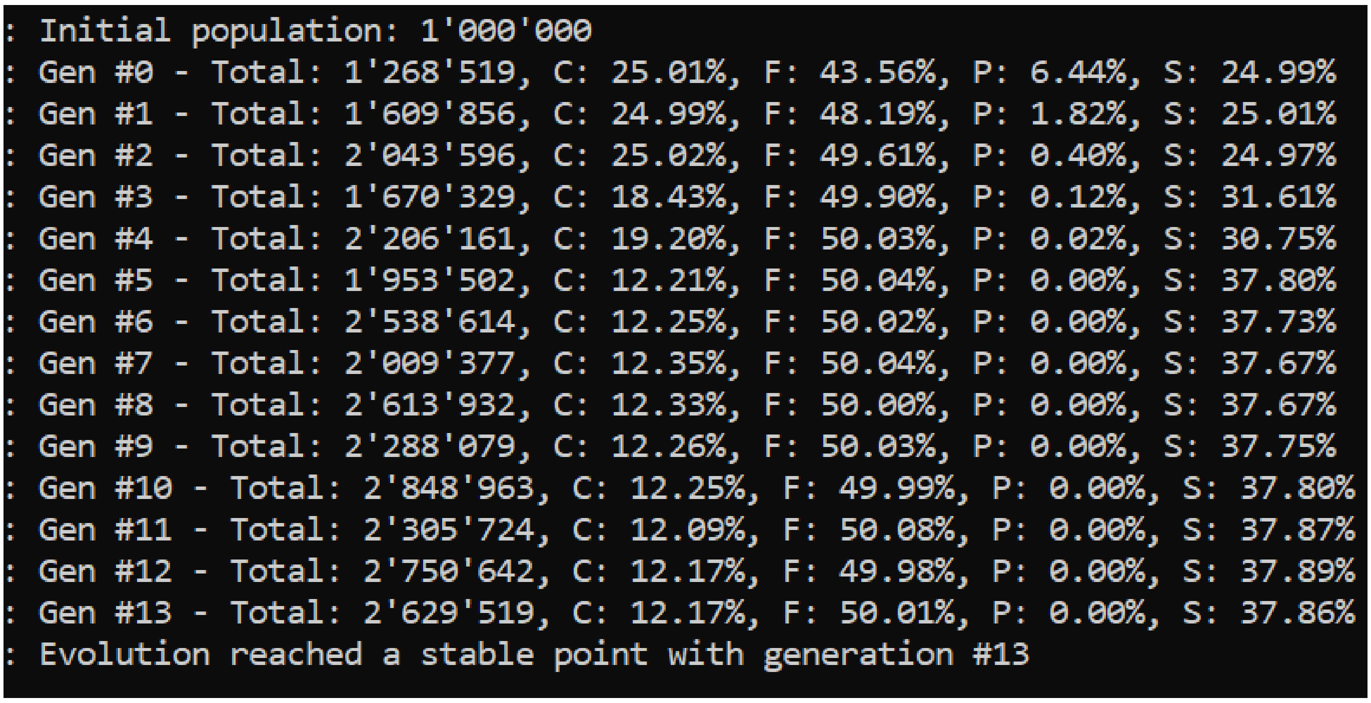
4. Case Studies

4.1. Default Settings

a	b	c
15	20	3

These are the settings written by Dawkins; we observe that, with our model, Philanderers die out, while coy remain stable as one third of the Fast.

This happens because Philanderers are not preferred from both types of women, and Fast women are more desirable (based on the payoffs).



```
: Initial population: 1'000'000
: Gen #0 - Total: 1'268'519, C: 25.01%, F: 43.56%, P: 6.44%, S: 24.99%
: Gen #1 - Total: 1'609'856, C: 24.99%, F: 48.19%, P: 1.82%, S: 25.01%
: Gen #2 - Total: 2'043'596, C: 25.02%, F: 49.61%, P: 0.40%, S: 24.97%
: Gen #3 - Total: 1'670'329, C: 18.43%, F: 49.90%, P: 0.12%, S: 31.61%
: Gen #4 - Total: 2'206'161, C: 19.20%, F: 50.03%, P: 0.02%, S: 30.75%
: Gen #5 - Total: 1'953'502, C: 12.21%, F: 50.04%, P: 0.00%, S: 37.80%
: Gen #6 - Total: 2'538'614, C: 12.25%, F: 50.02%, P: 0.00%, S: 37.73%
: Gen #7 - Total: 2'009'377, C: 12.35%, F: 50.04%, P: 0.00%, S: 37.67%
: Gen #8 - Total: 2'613'932, C: 12.33%, F: 50.00%, P: 0.00%, S: 37.67%
: Gen #9 - Total: 2'288'079, C: 12.26%, F: 50.03%, P: 0.00%, S: 37.75%
: Gen #10 - Total: 2'848'963, C: 12.25%, F: 49.99%, P: 0.00%, S: 37.80%
: Gen #11 - Total: 2'305'724, C: 12.09%, F: 50.08%, P: 0.00%, S: 37.87%
: Gen #12 - Total: 2'750'642, C: 12.17%, F: 49.98%, P: 0.00%, S: 37.89%
: Gen #13 - Total: 2'629'519, C: 12.17%, F: 50.01%, P: 0.00%, S: 37.86%
: Evolution reached a stable point with generation #13
```

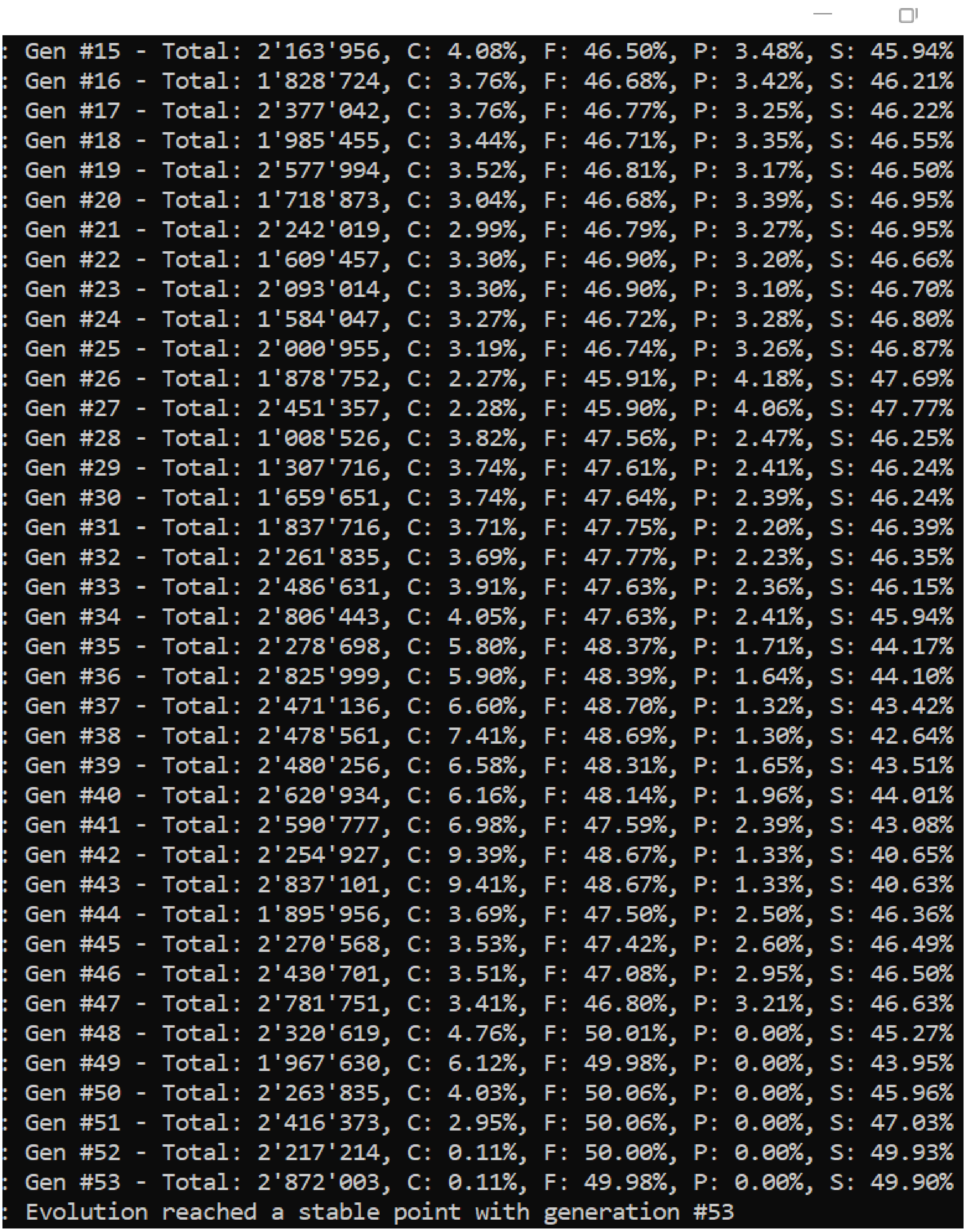
4.2. All Parameters to 0

a	b	c
0	0	0

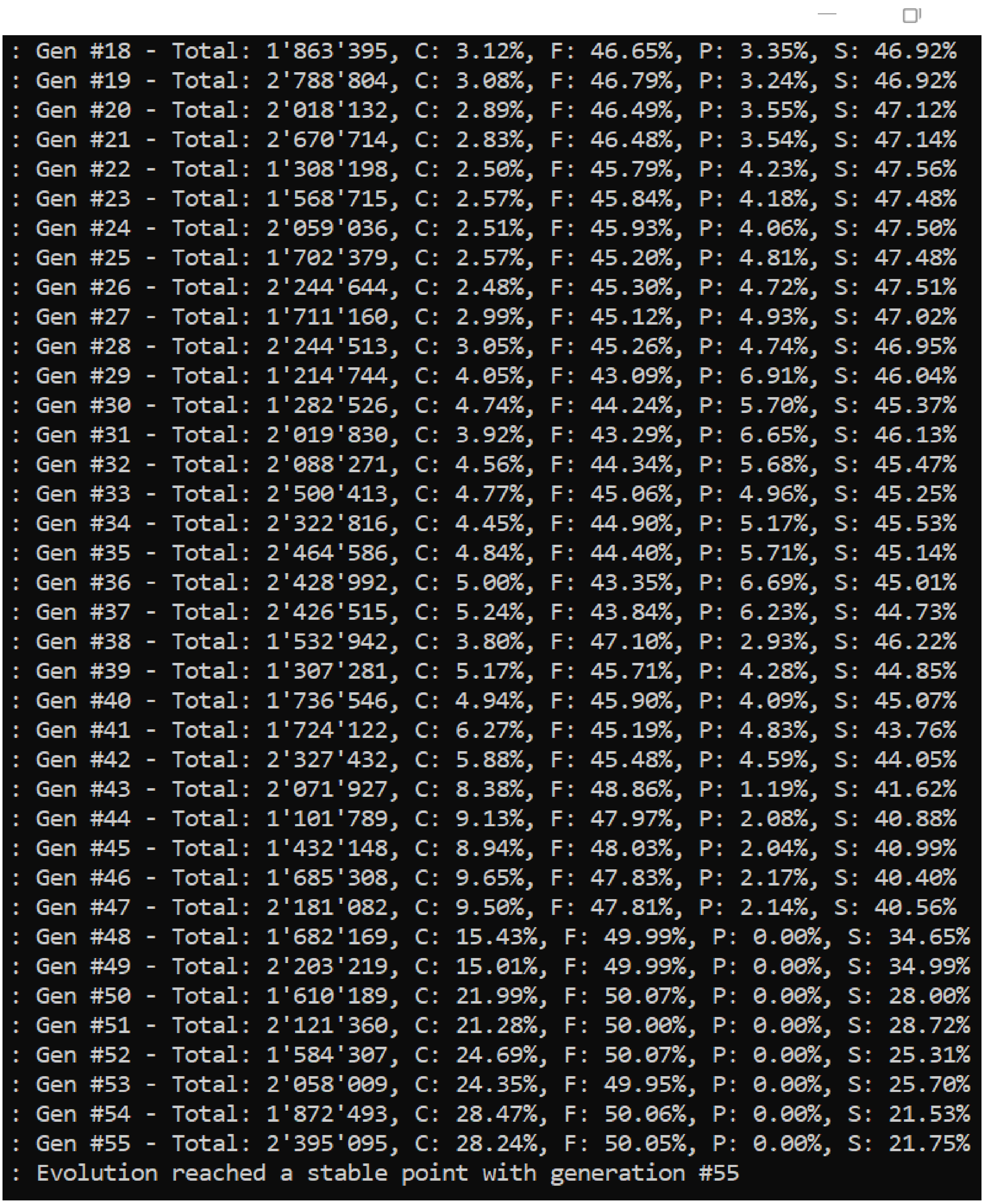
For these values, the matching is *Random*, in the sense that no one has any incentive in choosing one specific partner.

The interesting result is that, while the proportions between women types can vary, for men the situation is different: Philanderers always die out.

Probably this is due to their inherent limit of procreating only with Fast women.



```
: Gen #15 - Total: 2'163'956, C: 4.08%, F: 46.50%, P: 3.48%, S: 45.94%
: Gen #16 - Total: 1'828'724, C: 3.76%, F: 46.68%, P: 3.42%, S: 46.21%
: Gen #17 - Total: 2'377'042, C: 3.76%, F: 46.77%, P: 3.25%, S: 46.22%
: Gen #18 - Total: 1'985'455, C: 3.44%, F: 46.71%, P: 3.35%, S: 46.55%
: Gen #19 - Total: 2'577'994, C: 3.52%, F: 46.81%, P: 3.17%, S: 46.50%
: Gen #20 - Total: 1'718'873, C: 3.04%, F: 46.68%, P: 3.39%, S: 46.95%
: Gen #21 - Total: 2'242'019, C: 2.99%, F: 46.79%, P: 3.27%, S: 46.95%
: Gen #22 - Total: 1'609'457, C: 3.30%, F: 46.90%, P: 3.20%, S: 46.66%
: Gen #23 - Total: 2'093'014, C: 3.30%, F: 46.90%, P: 3.10%, S: 46.70%
: Gen #24 - Total: 1'584'047, C: 3.27%, F: 46.72%, P: 3.28%, S: 46.80%
: Gen #25 - Total: 2'000'955, C: 3.19%, F: 46.74%, P: 3.26%, S: 46.87%
: Gen #26 - Total: 1'878'752, C: 2.27%, F: 45.91%, P: 4.18%, S: 47.69%
: Gen #27 - Total: 2'451'357, C: 2.28%, F: 45.90%, P: 4.06%, S: 47.77%
: Gen #28 - Total: 1'008'526, C: 3.82%, F: 47.56%, P: 2.47%, S: 46.25%
: Gen #29 - Total: 1'307'716, C: 3.74%, F: 47.61%, P: 2.41%, S: 46.24%
: Gen #30 - Total: 1'659'651, C: 3.74%, F: 47.64%, P: 2.39%, S: 46.24%
: Gen #31 - Total: 1'837'716, C: 3.71%, F: 47.75%, P: 2.20%, S: 46.39%
: Gen #32 - Total: 2'261'835, C: 3.69%, F: 47.77%, P: 2.23%, S: 46.35%
: Gen #33 - Total: 2'486'631, C: 3.91%, F: 47.63%, P: 2.36%, S: 46.15%
: Gen #34 - Total: 2'806'443, C: 4.05%, F: 47.63%, P: 2.41%, S: 45.94%
: Gen #35 - Total: 2'278'698, C: 5.80%, F: 48.37%, P: 1.71%, S: 44.17%
: Gen #36 - Total: 2'825'999, C: 5.90%, F: 48.39%, P: 1.64%, S: 44.10%
: Gen #37 - Total: 2'471'136, C: 6.60%, F: 48.70%, P: 1.32%, S: 43.42%
: Gen #38 - Total: 2'478'561, C: 7.41%, F: 48.69%, P: 1.30%, S: 42.64%
: Gen #39 - Total: 2'480'256, C: 6.58%, F: 48.31%, P: 1.65%, S: 43.51%
: Gen #40 - Total: 2'620'934, C: 6.16%, F: 48.14%, P: 1.96%, S: 44.01%
: Gen #41 - Total: 2'590'777, C: 6.98%, F: 47.59%, P: 2.39%, S: 43.08%
: Gen #42 - Total: 2'254'927, C: 9.39%, F: 48.67%, P: 1.33%, S: 40.65%
: Gen #43 - Total: 2'837'101, C: 9.41%, F: 48.67%, P: 1.33%, S: 40.63%
: Gen #44 - Total: 1'895'956, C: 3.69%, F: 47.50%, P: 2.50%, S: 46.36%
: Gen #45 - Total: 2'270'568, C: 3.53%, F: 47.42%, P: 2.60%, S: 46.49%
: Gen #46 - Total: 2'430'701, C: 3.51%, F: 47.08%, P: 2.95%, S: 46.50%
: Gen #47 - Total: 2'781'751, C: 3.41%, F: 46.80%, P: 3.21%, S: 46.63%
: Gen #48 - Total: 2'320'619, C: 4.76%, F: 50.01%, P: 0.00%, S: 45.27%
: Gen #49 - Total: 1'967'630, C: 6.12%, F: 49.98%, P: 0.00%, S: 43.95%
: Gen #50 - Total: 2'263'835, C: 4.03%, F: 50.06%, P: 0.00%, S: 45.96%
: Gen #51 - Total: 2'416'373, C: 2.95%, F: 50.06%, P: 0.00%, S: 47.03%
: Gen #52 - Total: 2'217'214, C: 0.11%, F: 50.00%, P: 0.00%, S: 49.93%
: Gen #53 - Total: 2'872'003, C: 0.11%, F: 49.98%, P: 0.00%, S: 49.90%
: Evolution reached a stable point with generation #53
```



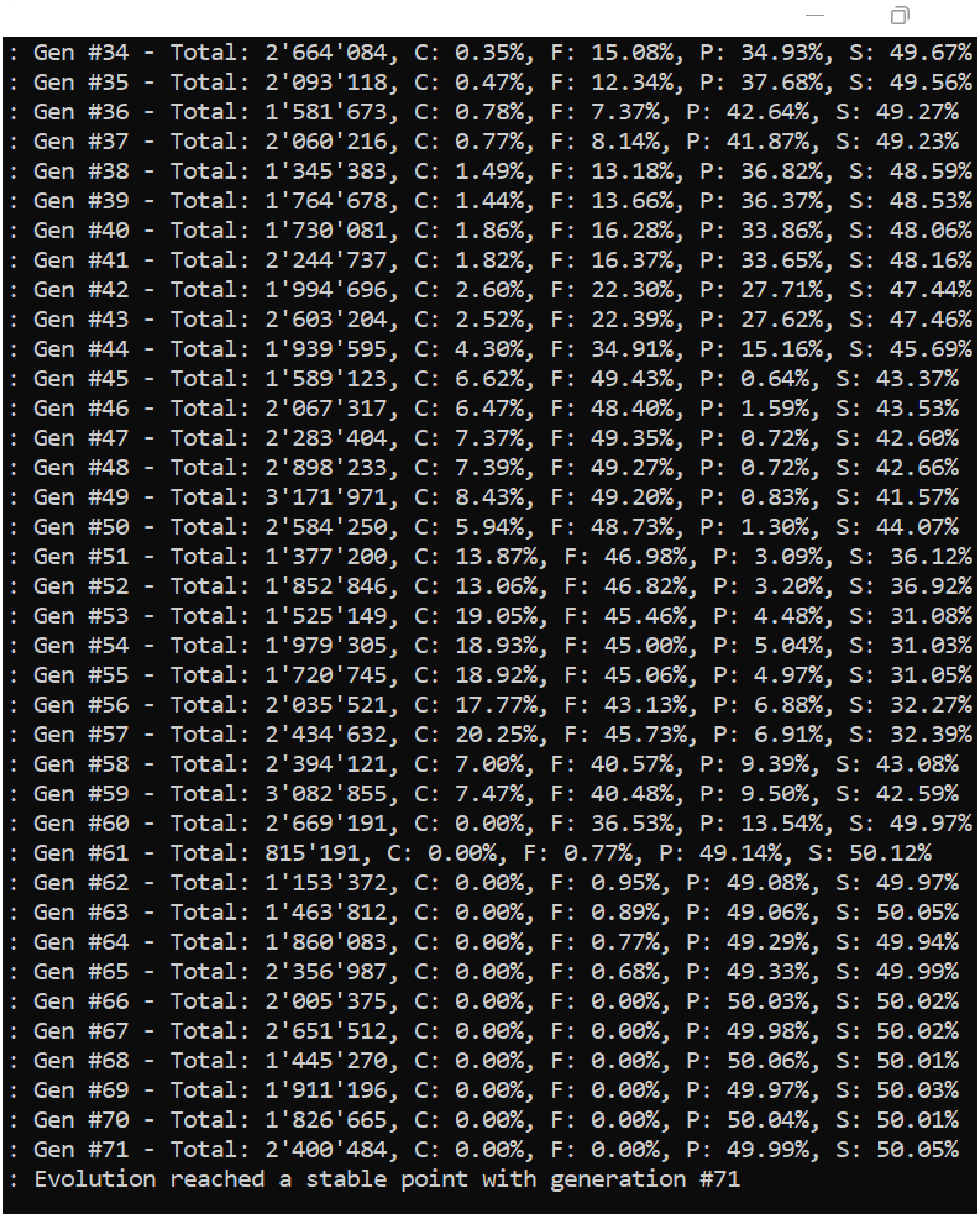
```
: Gen #18 - Total: 1'863'395, C: 3.12%, F: 46.65%, P: 3.35%, S: 46.92%
: Gen #19 - Total: 2'788'804, C: 3.08%, F: 46.79%, P: 3.24%, S: 46.92%
: Gen #20 - Total: 2'018'132, C: 2.89%, F: 46.49%, P: 3.55%, S: 47.12%
: Gen #21 - Total: 2'670'714, C: 2.83%, F: 46.48%, P: 3.54%, S: 47.14%
: Gen #22 - Total: 1'308'198, C: 2.50%, F: 45.79%, P: 4.23%, S: 47.56%
: Gen #23 - Total: 1'568'715, C: 2.57%, F: 45.84%, P: 4.18%, S: 47.48%
: Gen #24 - Total: 2'059'036, C: 2.51%, F: 45.93%, P: 4.06%, S: 47.50%
: Gen #25 - Total: 1'702'379, C: 2.57%, F: 45.20%, P: 4.81%, S: 47.48%
: Gen #26 - Total: 2'244'644, C: 2.48%, F: 45.30%, P: 4.72%, S: 47.51%
: Gen #27 - Total: 1'711'160, C: 2.99%, F: 45.12%, P: 4.93%, S: 47.02%
: Gen #28 - Total: 2'244'513, C: 3.05%, F: 45.26%, P: 4.74%, S: 46.95%
: Gen #29 - Total: 1'214'744, C: 4.05%, F: 43.09%, P: 6.91%, S: 46.04%
: Gen #30 - Total: 1'282'526, C: 4.74%, F: 44.24%, P: 5.70%, S: 45.37%
: Gen #31 - Total: 2'019'830, C: 3.92%, F: 43.29%, P: 6.65%, S: 46.13%
: Gen #32 - Total: 2'088'271, C: 4.56%, F: 44.34%, P: 5.68%, S: 45.47%
: Gen #33 - Total: 2'500'413, C: 4.77%, F: 45.06%, P: 4.96%, S: 45.25%
: Gen #34 - Total: 2'322'816, C: 4.45%, F: 44.90%, P: 5.17%, S: 45.53%
: Gen #35 - Total: 2'464'586, C: 4.84%, F: 44.40%, P: 5.71%, S: 45.14%
: Gen #36 - Total: 2'428'992, C: 5.00%, F: 43.35%, P: 6.69%, S: 45.01%
: Gen #37 - Total: 2'426'515, C: 5.24%, F: 43.84%, P: 6.23%, S: 44.73%
: Gen #38 - Total: 1'532'942, C: 3.80%, F: 47.10%, P: 2.93%, S: 46.22%
: Gen #39 - Total: 1'307'281, C: 5.17%, F: 45.71%, P: 4.28%, S: 44.85%
: Gen #40 - Total: 1'736'546, C: 4.94%, F: 45.90%, P: 4.09%, S: 45.07%
: Gen #41 - Total: 1'724'122, C: 6.27%, F: 45.19%, P: 4.83%, S: 43.76%
: Gen #42 - Total: 2'327'432, C: 5.88%, F: 45.48%, P: 4.59%, S: 44.05%
: Gen #43 - Total: 2'071'927, C: 8.38%, F: 48.86%, P: 1.19%, S: 41.62%
: Gen #44 - Total: 1'101'789, C: 9.13%, F: 47.97%, P: 2.08%, S: 40.88%
: Gen #45 - Total: 1'432'148, C: 8.94%, F: 48.03%, P: 2.04%, S: 40.99%
: Gen #46 - Total: 1'685'308, C: 9.65%, F: 47.83%, P: 2.17%, S: 40.40%
: Gen #47 - Total: 2'181'082, C: 9.50%, F: 47.81%, P: 2.14%, S: 40.56%
: Gen #48 - Total: 1'682'169, C: 15.43%, F: 49.99%, P: 0.00%, S: 34.65%
: Gen #49 - Total: 2'203'219, C: 15.01%, F: 49.99%, P: 0.00%, S: 34.99%
: Gen #50 - Total: 1'610'189, C: 21.99%, F: 50.07%, P: 0.00%, S: 28.00%
: Gen #51 - Total: 2'121'360, C: 21.28%, F: 50.00%, P: 0.00%, S: 28.72%
: Gen #52 - Total: 1'584'307, C: 24.69%, F: 50.07%, P: 0.00%, S: 25.31%
: Gen #53 - Total: 2'058'009, C: 24.35%, F: 49.95%, P: 0.00%, S: 25.70%
: Gen #54 - Total: 1'872'493, C: 28.47%, F: 50.06%, P: 0.00%, S: 21.53%
: Gen #55 - Total: 2'395'095, C: 28.24%, F: 50.05%, P: 0.00%, S: 21.75%
: Evolution reached a stable point with generation #55
```

4.3. Negative Parameter b

a	**b**	**c**
15	-20	3

Here we made the **b** parameter negative, so that it changes all the preferences.

Now Fast women prefer Philanderers. Faithful and Coy die out.



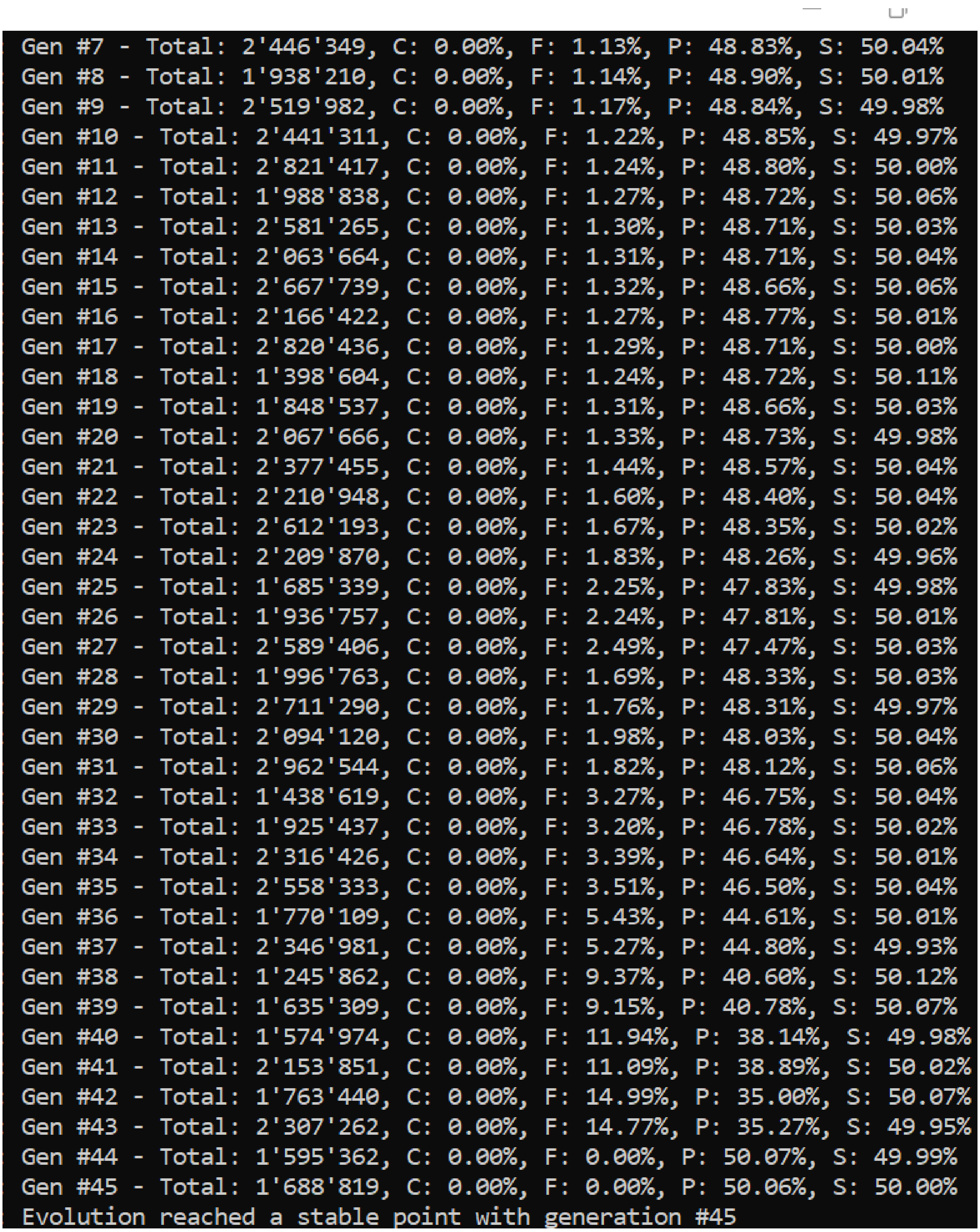
```
: Gen #34 - Total: 2'664'084, C: 0.35%, F: 15.08%, P: 34.93%, S: 49.67%
: Gen #35 - Total: 2'093'118, C: 0.47%, F: 12.34%, P: 37.68%, S: 49.56%
: Gen #36 - Total: 1'581'673, C: 0.78%, F: 7.37%, P: 42.64%, S: 49.27%
: Gen #37 - Total: 2'060'216, C: 0.77%, F: 8.14%, P: 41.87%, S: 49.23%
: Gen #38 - Total: 1'345'383, C: 1.49%, F: 13.18%, P: 36.82%, S: 48.59%
: Gen #39 - Total: 1'764'678, C: 1.44%, F: 13.66%, P: 36.37%, S: 48.53%
: Gen #40 - Total: 1'730'081, C: 1.86%, F: 16.28%, P: 33.86%, S: 48.06%
: Gen #41 - Total: 2'244'737, C: 1.82%, F: 16.37%, P: 33.65%, S: 48.16%
: Gen #42 - Total: 1'994'696, C: 2.60%, F: 22.30%, P: 27.71%, S: 47.44%
: Gen #43 - Total: 2'603'204, C: 2.52%, F: 22.39%, P: 27.62%, S: 47.46%
: Gen #44 - Total: 1'939'595, C: 4.30%, F: 34.91%, P: 15.16%, S: 45.69%
: Gen #45 - Total: 1'589'123, C: 6.62%, F: 49.43%, P: 0.64%, S: 43.37%
: Gen #46 - Total: 2'067'317, C: 6.47%, F: 48.40%, P: 1.59%, S: 43.53%
: Gen #47 - Total: 2'283'404, C: 7.37%, F: 49.35%, P: 0.72%, S: 42.60%
: Gen #48 - Total: 2'898'233, C: 7.39%, F: 49.27%, P: 0.72%, S: 42.66%
: Gen #49 - Total: 3'171'971, C: 8.43%, F: 49.20%, P: 0.83%, S: 41.57%
: Gen #50 - Total: 2'584'250, C: 5.94%, F: 48.73%, P: 1.30%, S: 44.07%
: Gen #51 - Total: 1'377'200, C: 13.87%, F: 46.98%, P: 3.09%, S: 36.12%
: Gen #52 - Total: 1'852'846, C: 13.06%, F: 46.82%, P: 3.20%, S: 36.92%
: Gen #53 - Total: 1'525'149, C: 19.05%, F: 45.46%, P: 4.48%, S: 31.08%
: Gen #54 - Total: 1'979'305, C: 18.93%, F: 45.00%, P: 5.04%, S: 31.03%
: Gen #55 - Total: 1'720'745, C: 18.92%, F: 45.06%, P: 4.97%, S: 31.05%
: Gen #56 - Total: 2'035'521, C: 17.77%, F: 43.13%, P: 6.88%, S: 32.27%
: Gen #57 - Total: 2'434'632, C: 20.25%, F: 45.73%, P: 6.91%, S: 32.39%
: Gen #58 - Total: 2'394'121, C: 7.00%, F: 40.57%, P: 9.39%, S: 43.08%
: Gen #59 - Total: 3'082'855, C: 7.47%, F: 40.48%, P: 9.50%, S: 42.59%
: Gen #60 - Total: 2'669'191, C: 0.00%, F: 36.53%, P: 13.54%, S: 49.97%
: Gen #61 - Total: 815'191, C: 0.00%, F: 0.77%, P: 49.14%, S: 50.12%
: Gen #62 - Total: 1'153'372, C: 0.00%, F: 0.95%, P: 49.08%, S: 49.97%
: Gen #63 - Total: 1'463'812, C: 0.00%, F: 0.89%, P: 49.06%, S: 50.05%
: Gen #64 - Total: 1'860'083, C: 0.00%, F: 0.77%, P: 49.29%, S: 49.94%
: Gen #65 - Total: 2'356'987, C: 0.00%, F: 0.68%, P: 49.33%, S: 49.99%
: Gen #66 - Total: 2'005'375, C: 0.00%, F: 0.00%, P: 50.03%, S: 50.02%
: Gen #67 - Total: 2'651'512, C: 0.00%, F: 0.00%, P: 49.98%, S: 50.02%
: Gen #68 - Total: 1'445'270, C: 0.00%, F: 0.00%, P: 50.06%, S: 50.01%
: Gen #69 - Total: 1'911'196, C: 0.00%, F: 0.00%, P: 49.97%, S: 50.03%
: Gen #70 - Total: 1'826'665, C: 0.00%, F: 0.00%, P: 50.04%, S: 50.01%
: Gen #71 - Total: 2'400'484, C: 0.00%, F: 0.00%, P: 49.99%, S: 50.05%
: Evolution reached a stable point with generation #71
```

4.4. 99% Philanderers

a	**b**	**c**	**P**	**F**
15	20	3	99%	1%

The evolutionary parameters are the same, what changes is the percentage of starting population.

We see how Faithful begin to grow in population but the majority of philanderers, even if not mostly desirable, defeats them.



```
Gen #7 - Total: 2'446'349, C: 0.00%, F: 1.13%, P: 48.83%, S: 50.04%
Gen #8 - Total: 1'938'210, C: 0.00%, F: 1.14%, P: 48.90%, S: 50.01%
Gen #9 - Total: 2'519'982, C: 0.00%, F: 1.17%, P: 48.84%, S: 49.98%
Gen #10 - Total: 2'441'311, C: 0.00%, F: 1.22%, P: 48.85%, S: 49.97%
Gen #11 - Total: 2'821'417, C: 0.00%, F: 1.24%, P: 48.80%, S: 50.00%
Gen #12 - Total: 1'988'838, C: 0.00%, F: 1.27%, P: 48.72%, S: 50.06%
Gen #13 - Total: 2'581'265, C: 0.00%, F: 1.30%, P: 48.71%, S: 50.03%
Gen #14 - Total: 2'063'664, C: 0.00%, F: 1.31%, P: 48.71%, S: 50.04%
Gen #15 - Total: 2'667'739, C: 0.00%, F: 1.32%, P: 48.66%, S: 50.06%
Gen #16 - Total: 2'166'422, C: 0.00%, F: 1.27%, P: 48.77%, S: 50.01%
Gen #17 - Total: 2'820'436, C: 0.00%, F: 1.29%, P: 48.71%, S: 50.00%
Gen #18 - Total: 1'398'604, C: 0.00%, F: 1.24%, P: 48.72%, S: 50.11%
Gen #19 - Total: 1'848'537, C: 0.00%, F: 1.31%, P: 48.66%, S: 50.03%
Gen #20 - Total: 2'067'666, C: 0.00%, F: 1.33%, P: 48.73%, S: 49.98%
Gen #21 - Total: 2'377'455, C: 0.00%, F: 1.44%, P: 48.57%, S: 50.04%
Gen #22 - Total: 2'210'948, C: 0.00%, F: 1.60%, P: 48.40%, S: 50.04%
Gen #23 - Total: 2'612'193, C: 0.00%, F: 1.67%, P: 48.35%, S: 50.02%
Gen #24 - Total: 2'209'870, C: 0.00%, F: 1.83%, P: 48.26%, S: 49.96%
Gen #25 - Total: 1'685'339, C: 0.00%, F: 2.25%, P: 47.83%, S: 49.98%
Gen #26 - Total: 1'936'757, C: 0.00%, F: 2.24%, P: 47.81%, S: 50.01%
Gen #27 - Total: 2'589'406, C: 0.00%, F: 2.49%, P: 47.47%, S: 50.03%
Gen #28 - Total: 1'996'763, C: 0.00%, F: 1.69%, P: 48.33%, S: 50.03%
Gen #29 - Total: 2'711'290, C: 0.00%, F: 1.76%, P: 48.31%, S: 49.97%
Gen #30 - Total: 2'094'120, C: 0.00%, F: 1.98%, P: 48.03%, S: 50.04%
Gen #31 - Total: 2'962'544, C: 0.00%, F: 1.82%, P: 48.12%, S: 50.06%
Gen #32 - Total: 1'438'619, C: 0.00%, F: 3.27%, P: 46.75%, S: 50.04%
Gen #33 - Total: 1'925'437, C: 0.00%, F: 3.20%, P: 46.78%, S: 50.02%
Gen #34 - Total: 2'316'426, C: 0.00%, F: 3.39%, P: 46.64%, S: 50.01%
Gen #35 - Total: 2'558'333, C: 0.00%, F: 3.51%, P: 46.50%, S: 50.04%
Gen #36 - Total: 1'770'109, C: 0.00%, F: 5.43%, P: 44.61%, S: 50.01%
Gen #37 - Total: 2'346'981, C: 0.00%, F: 5.27%, P: 44.80%, S: 49.93%
Gen #38 - Total: 1'245'862, C: 0.00%, F: 9.37%, P: 40.60%, S: 50.12%
Gen #39 - Total: 1'635'309, C: 0.00%, F: 9.15%, P: 40.78%, S: 50.07%
Gen #40 - Total: 1'574'974, C: 0.00%, F: 11.94%, P: 38.14%, S: 49.98%
Gen #41 - Total: 2'153'851, C: 0.00%, F: 11.09%, P: 38.89%, S: 50.02%
Gen #42 - Total: 1'763'440, C: 0.00%, F: 14.99%, P: 35.00%, S: 50.07%
Gen #43 - Total: 2'307'262, C: 0.00%, F: 14.77%, P: 35.27%, S: 49.95%
Gen #44 - Total: 1'595'362, C: 0.00%, F: 0.00%, P: 50.07%, S: 49.99%
Gen #45 - Total: 1'688'819, C: 0.00%, F: 0.00%, P: 50.06%, S: 50.00%
Evolution reached a stable point with generation #45
```