Simulation of Spatial Self-Organization in a Stepping Stone Environment

Bilal Gonen[1], **Guy Hoelzer**[2]
[1]Computer Science and Engineering Department, University of Nevada, Reno, Reno, Nevada, U.S.A.
[2]Biology Department, University of Nevada, Reno, Reno, Nevada, U.S.A.

Abstract— *Self-organization is the process where a structure or pattern appears in a system without a central authority or external element imposing it through planning. In various areas, simulation is a safe and low-cost alternative to complex problems. We developed a free online simulation tool which computes and displays spatial self-organizing behavior of individuals in a stepping stone environment.*

Keywords: bioinformatics, simulation, spatial self-organization

1. Introduction

Self-organization is the process where a structure or pattern appears in a system without a central authority or external element imposing it through planning. This globally coherent pattern appears from the local interaction of the elements that make up the system, thus the organization is achieved in a way that is parallel (all the elements act at the same time) and distributed (no element is a coordinator) [1].

The concept of self-organization is central to the description of biological systems, from the subcellular to the ecosystem level. There are also cited examples of "self-organizing" behaviour found in the literature of many other disciplines, both in the natural sciences and the social sciences such as economics or anthropology [2] [3] [4] [5] [6] [7]. Self-organization has also been observed in mathematical systems such as cellular automata. Sometimes the notion of self-organization is conflated with that of the related concept of emergence. The link between emergence and self-organization remains an active research question [1].

Originally, the term "self-organizing" was used by Immanuel Kant in his Critique of Judgment [8], where he argued that teleology is a meaningful concept only if there exists such an entity whose parts or "organs" are simultaneously ends and means. Such a system of organs must be able to behave as if it has a mind of its own, that is, it is capable of governing itself [1].

The idea that the dynamics of a system can tend by itself to increase the inherent order of a system has a long history. One of the earliest statements of this idea was by the philosopher Descartes, in the fifth part of his Discourse on Method [9], where he presents it hypothetically [1].

According to Scott Camazine, in biological systems self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern [10].

In various areas, simulation is a safe and low-cost alternative to complex problems. We developed a free online simulation tool which computes and displays spatial self-organizing behavior of individuals in a stepping stone environment. For the public consumption by the research community, we made it a free online simulator at `www.evosimulator.com`. The simulator is developed in C#, .Net, and Silverlight 4.0 to provide more visual insight to the users. In this paper, we use a real-world example. We simulated "Beetle World" which is an experiment conducted in our biology department. In this experiment, plates are used as stepping stones, and beetles are used as individuals. First, Section-2 gives an overview of the steps in the simulation. Then, the online simulation tool is described in Section-3.

2. EvoSimulator

In this section, the EvoSimulator algorithm is explained in detail. The program is developed in object-oriented approach. There are five classes used in this simulator, i.e., Generation, Grouplist, Group, Plate, and Beetle. There are several user-inputted parameters for this simulation. These parameters are;

- Number of plates
- Number of beetles per plate
- Number of alleles
- Number of generations

When we started building this simulation tool, we were simulating an experiment where beetles move between plates. In that experiment, plates represent stepping stones. In each generation, for the next offspring, to ensure the fixed amount of beetles to be produced in each plate, the same amount of beans are put into each plate. This ensures that the same number of offspring to be produced in each plate, since one bean can hold at most one beetle egg. Each plate is connected to two other plates. We also connect the first and last plates to each other forming a circular linear shape. Number of alleles is also a user-inputted parameter for our simulation. The beetles inherit their chromosome

from their mother. Algorithm1 shows the main steps in EvoSimulator algorithm.

Algorithm 1 $EvoSimulator\_main\_function$

```
Create beetles and position plates
Call splitMaster function
for genID = 1 to numOfGenerations do
    Create an object of Generation class;
    Call makeChildrenForAllPlates function;
    Call fillVacancies function;
    Call killParentBeetles function;
    Call splitMaster function;
end for
```

2.1 Creating Beetles and Positioning Plates

The plates in the simulation are placed in a circular chain shape. Each plate has two neighbor plates. N being the number of plates, to make it circular shape, plate-1 is neighbor to plate-2 and plate-N. Then we create beetles. We set the gender of beetles to either male or female with equal probability. Each beetle has a unique ID number assigned to them so that we can keep track of them during the simulation. Each beetle has also an allele assigned to them when they are created. We also set the number of children that a beetle will produce during its life span. This number is produced by using poisson distribution. As the lambda parameter of poisson distribution, we used 1.8 in our simulation. We then place the beetles in plates in equal quantity. This number of beetles per plate is set by user before the simulation begins.

2.2 Grouping The Plates

We split the plates into subdivisions based on their FST values. FST (Fixation index) is a measure of population differentiation, genetic distance, based on genetic polymorphism data. It is simply the correlation of randomly chosen alleles within the same sub-population relative to that found in the entire population. It is often expressed as the proportion of genetic diversity due to allele frequency differences among populations [1].

As the grouping heuristic, we used a top-down approach. That is, we put all the plates into one group, and begin splitting the groups into subgroups. In each generation, we create an empty group and we put all the plates into that group. Then, we call the splitMaster function which is illustrated in algorithm-2. In this function, there are two types of functions; (i) Analyze function which computes from where to split the group that maximizes the overall FST. (ii) Split function that actually splits the group into two groups. Based on the number of groups in the system, these two functions operate differently. The reason is that the plates are placed in a circular shape. We need to compute not only from which plate to split the group, but also from which plate should the first group start. Algorithm-3 is the pseudo-code of the splitTheOnlyGroup function. This process is displayed in figure 1. We set the minimum size for a group as 2. That is there must be at least two plates in a group. The AnalyzeTheOnlyGroup function finds both first-cut and split points which gives the maximum FST. After the group is split into two groups, the splitMaster function calls the splitGroupListWhenMultipleGroups function. This function iterates through all the groups, and within each group it checks each possible split point to find out which group to split and which point to split that maximizes the overall FST. Algorithm-4 is the pseudo-code of the splitGroupList-WhenMultipleGroups function. And the process of splitting grouplist when multiple groups is illusrated in figure 2.

Algorithm 2 $splitMaster$

```
while groups still can be splitted do
    if there is only one group then
        Call AnalyzeTheOnlyGroup function;
        Call splitTheOnlyGroup function;
    else
        Call AnalyzeMultipleGroups function;
        Call splitGroupListWhenMultipleGroups function;
    end if
end while
```

Algorithm 3 $splitTheOnlyGroup$

```
for first_cut = 0 to numOfPlates do
    calculate fromPlateIndex;
    calculate toPlateIndex;
    for i = fromPlateIndex to toPlateIndex do
        Find a split point that maximizes the FST
    end for
end for
```

Algorithm 4 $splitGroupListWhenMultipleGroups$

```
for group_ID = 0 to numOfGroups do
    if plateCountInThisGroup > minGroupSize + 1
    then
        Find a split point in group that maximizes FST
    end if
end for
```

2.3 Creating New Generations

There are three main steps happening in each generation. Those are;

- Produce offspring and put them into plates.
- Fill Vacancies in the plates.

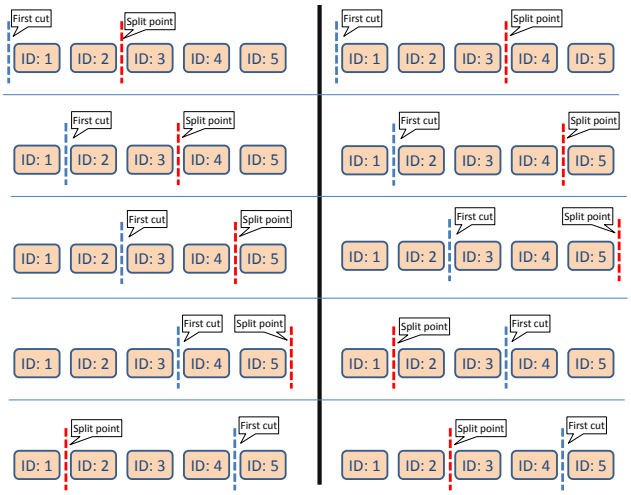

Fig. 1: Process of splitting the grouplist initially

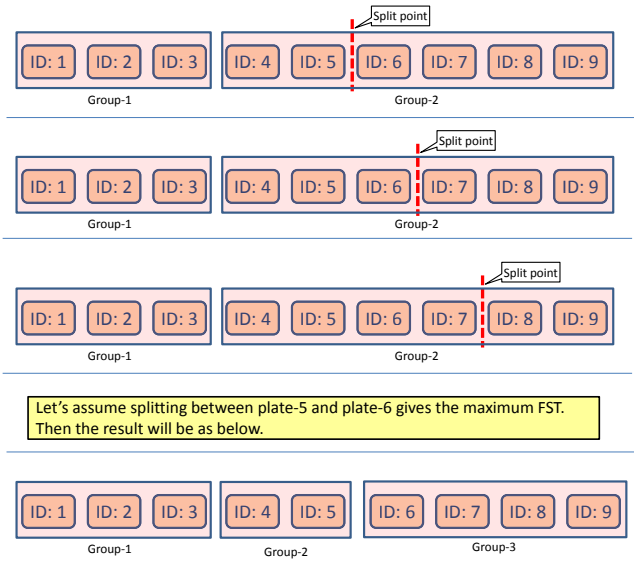

Fig. 2: Process of splitting grouplist when multiple groups

- Kill parent individuals.

2.3.1 Produce Offspring and Put Them Into Plates

In the simulation, we check every beetles in every plates. If the beetle is a female, then we get the number of children to produce from that female beetle. The number of children is generated for each female beetle by using poisson distribution. And this number is assigned to them when the beetles are created. Before producing any child from that parent, we make sure if there is some vacant spot in any plate around the female parent to place her child. We use Gaussian distribution to determine how far away the child will be from their mother. After determining which plate to put the child, we check the plate to see if the plate has reached to its maximum number of children already. If

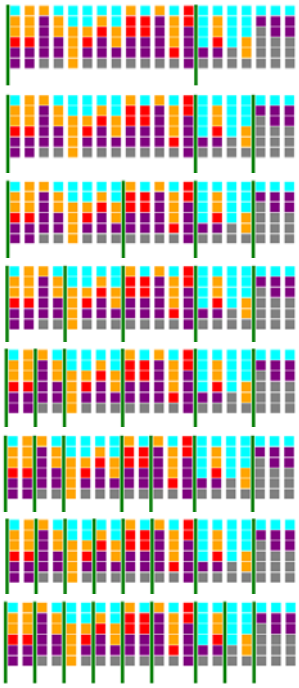

Fig. 3: groups

so, then we generate another moving distance number using normal distribution to put the child into another plate. We repeat this process until either we find a vacant spot in a plate, or we fail to find a vacant spot for some user-defined number of times. As long as we find vacant spot for the female parent to put her children, we keep repeating this process for the number of children to produce from that female parent. If we are able to find a vacant spot to place a child, then we produce a child from the parent and put the child into the plate. At the end of this repetition, if we are unable to find a vacant spot to place a child, then we don't generate any more child from the parent, and go to the next female parent in the plates.

2.3.2 Fill Vacancies in The Plates

After we fill in the plates with child beetles, there may be some vacant spots in the plates. To fill in those vacant spots in the plates, we try to find a female parent from other plates. We determine which plate to search by using gaussian distribution. When we find a female parent, we produce a child beetle from it, and put into the vacant spot. At the end of this process, all of the plates hold the same number of child beetle.

2.3.3 Kill Parent Beetles

In this step, we remove parent beetles from the plates. However, they still remain in the memory of the simulation with their ID numbers so that we can display some statistical

Fig. 4: Entry point for the parameters

information by using historical data.

3. EvoSimulator Tool

EvoSimulator is a free online simulation tool which computes and displays evolutionary behavior of individuals in a stepping stone environment. As mentioned above, there are several user-inputted parameters for this simulation. These parameters are; Number of plates, Number of beetles per plate, Number of alleles, and Number of generations. Figure-4 displays the entry point for these parameters in the simulator tool.

Figure-5 is a screenshot of the simulator after running it. The histogram shows that when the number of groups increases, the FST of the population also increases. Each vertical column below the histogram represents a plate, and each colored square represents a beetle. Colors of the beetles represent the chromosome type (alleles).

4. Conclusion

We developed a free online simulation tool to simulate spatial self-organizing behavior of individuals in a stepping stone environment. After several experiments, we observed that if we run the simulation long enough the spatial configuration of genetic variations in the population self-organizes.

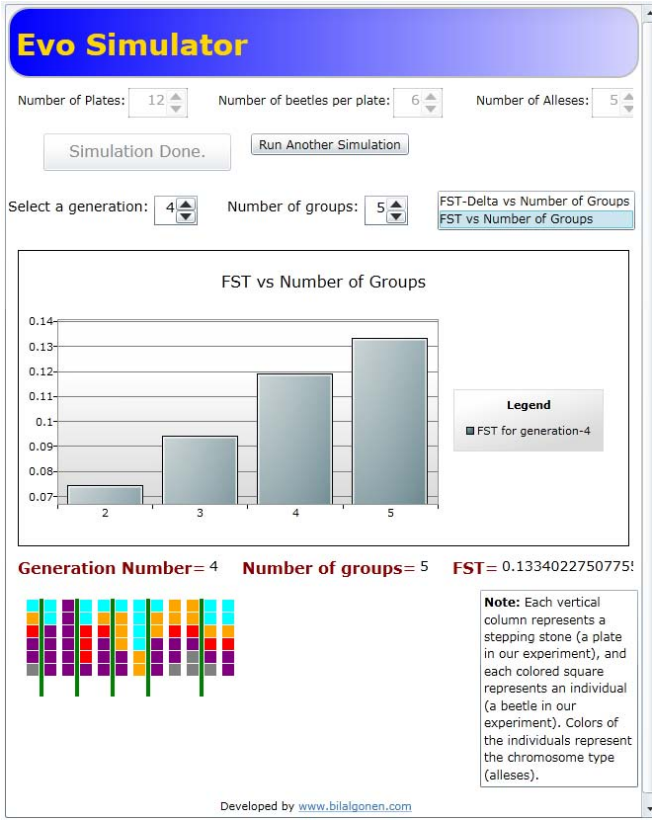

Fig. 5: A screenshot of the simulator after running it

References

[1] http://en.wikipedia.org.
[2] V. Castets, E. Dulos, J. Boissonade, and P. De Kepper, "Experimental evidence of a sustained standing turing-type nonequilibrium chemical pattern," *Phys. Rev. Lett.*, vol. 64, no. 24, pp. 2953–2956, Jun 1990.
[3] G. Nicolis and I. Prigogine, *Self-organization in nonequilibrium systems*. New York: Wiley, 1977.
[4] T. et al., *Proc. Natl. Acad. Sci. U.S.A. 99*, p. 9645, 2002.
[5] H. Wager, *Philos. Trans. R. Soc. London Ser. B 201*, p. 333, 1911.
[6] J. D. M. J. J. Tyson, *Development 106*, p. 421, 1989.
[7] S. M. Y. M. D. Bertness, S. D. Gaines, *Ecology 79*, p. 1382, 1998.
[8] I. Kant, *Critique of judgment*, 1987.
[9] "Rene descartes, discourse on method, 1637."
[10] S. Camazine, N. R. Franks, J. Sneyd, E. Bonabeau, J.-L. Deneubourg, and G. Theraula, *Self-Organization in Biological Systems*. Princeton, NJ, USA: Princeton University Press, 2001.