Finding my place in the crowd: Group formation and localization in a virtual environment

Andres Ruiz [1]

[1]University of New Mexico, Albuquerque, NM 87106
afruizc@unm.edu

Abstract

In the Movable Feast Machine, direct interaction is limited to small regions of space, but many computations could benefit from larger scale structure. This paper presents a simple aggregation and localization strategy allowing individuals to form communicating groups and find their own relative positions within the group. We demonstrate with different experiments that there is an *ideal* number of elements to achieve *fast* and *stable* organization and localization. We also propose an algorithm that, we conjecture, gives good consistence in the localization procedure and speeds up the process of total convergence of the universe.

Introduction

The Movable Feast Machine (MfM from now), is a platform that proposes a new paradigm in the way computation is performedAckley and Cannon (2011). In this paper we use this model and propose a new element whose behaviour uses the MfM architecture and extends it allowing new mechanics that are not possible with the mainstream flow of actions in a normal MfM application. The philosophy of the MfM architechture is being extremelly robust; *Robustness*, to give a compact, concise definition, is the ability to ***survive*** on several ***conditions***Silverman and Ikegami (2011). Notice that the bolded words have to be defined as well in a virtual environment, but we will just leave it up until here. *Robustness* also brings limitation with it; there is a sacrifice in performance and some lines are drawn: ensuring the completion of the program doesn't always go in agreement with conventional paradigms, which is why that some tasks that were trivially achievable before, now become imposibilities. We have indentified one of those crucial things, that once you enable it, computation evolves in an exponential way. We believe, that when computation becomes an spacial issue, when you implement the computation model into an spacial environment, space becomes a *big deal*. The MfM has a model in which a Von Neumman NeigborhoodToffoli and Margolus (1987) is provided to the atom that wakes up in an event, we are trying to make this neighborhood larger, by increasing it's size up to a certain point that is determined based on interaction between the atoms.

Specifically, The problem we are trying to address is to have blobs of atoms get together, *cluster* in different clumps. This is relevant because by having a set of atoms get close to each other, one can increase the power of computation, and perform more complex tasks. Given that computation is based on a spacial factor, specifically, interaction in the space is directly translated into computation, increasing the interaction (either directly or indirectly) between atoms, is directly translated into increasing of computation power, hence increasing the complexity of the tasks that can arise in this spatial setting, and giving birth to new opportunities that were not present before on the MfM architecture.

This paper provides a complete anaylisis of our protocol, along with several theoretical improvements, that once put in practice, we believe will boots the performance of both, the organization and the localization procedure. This document is organized as follows: the next section provides a detailed description of the protocol, the next section presents the experiments and the results we performed. Following that section, we present our conclussions and future work, and then we wrap it up with the bibliographic references.

Self-Organization and Localization

Self-organization and Localization are two concepts that will be extensively repeated from now on, which is why it is important to describe them as soon as possible. **Self-organzation** is the process of having a group of elements get together and form blobs that each have more than one atom. This will be described further down below, but it is sufficient to say that it is a process that will eventually conclude and the expected outcome is a number of blobs that depends on the number of initial atoms in the world, and their position in the universe. **Localization** is a process in which, after every atom is part of a blob[1], they start interacting and *talking* to each other in order to find where they are in the blob. Localization can be achieved in many ways, however, after having considered several alternatives, bounding rectangles and

[1]There is not a real checking or even ensuring of whether *every* atom forms part of a clump or not, it is something that happens based on the rules the ASDF atoms follow

directional counters were the best option. Bounding rectangles, as can be seen in figure 1, are a way of enclosing a blob within the tightest possible rectangle. Directional counters, as its name indicates, are counters for each one of four directions (North, South, East and West), keeping track of how many tiles there are between the atom and the bounding rectangle. To summarize, the self-organization process guarantees higher level objects being formed, while the localization one guarantees communication and synchronization in such objects.

Section 3.1: Model Overview

As mentioned earlier, the ASDF elements will self-organize in different blobs that will then start their localization process based on where they consider their relative position in the blob is. After several iterations of the localization process, convergence will be achieved and all the elements will then have an idea of where their place in the blob is. Their place will be determined by the amount of tiles that one element is away from the boundary elements. A boundary element is an element that is on the boundary and therefore helps forming the bounding rectangle, specifically, the a bounding element is an element such that at a certain coordinate (north, south, west or east) it doesnt have any ASDF neighbor elements towards that direction, i.e. all the neighbouring cells in that direction are empty. After the localization process, if there is some sort of interruption (like a nuke or some elements are moved by dreg), the elements would go back to the self-organization stage and then the localization process will start again.

We now briefly describe what an Event in the MfM architecture is because this will provide us with a useful framework to describe the behavior of the ASDF element in both, the organization and the localization stage. In its most simple form, an event is the means by which the mfm interacts with, and let's them know that it is their turn to wake up, interact with the environment, change its internal and its neighbors state and go back to sleep. An event is assigned at random to an event, and there is no guarantee on the amount of events an atom is receiving however, because of laws of probabilities and calculations it is true in practice that on average, all the elements will get an event after certain period of time.

Section 3.2. Model detailed description

We now give all the details on how both stages of the operations are performed, and we highlight the relevant points that allow our element to perform as expected.

Section 3.2.1: Self-organization process

The self-organization process is the basis that allows the final result to be achieved. The resulting state in which the elements are at the end of this stage, will determine how good or bad the localization process will turn out to be. The self-organization is summarized in algorithm 1.

Algorithm 1 Complete the self-organization process

```
1: procedure SELFORGANIZATION(())
2:     if I become attractor then
3:         $A[\,] =$ empty elements in EW
4:         $B[\,] =$ asdf elements in EW
5:         $i = j = 0$
6:         while (empty spots and elements to relocate) do
7:             swap $B[i]$with $A[j]$
8:             Turn $B[i]$ into attractor
9:             increment $i$ and $j$
```

When an atom is set to follow the self-organizing procedure, it starts at the beginning of an event. When the atom gets an event, it has some chances of becoming an *attractor* element. An attractor element is an element that, as it name suggests, takes elements that are around, puts them as close together as possible from where it is and then turns those elements into attractors themselves. If it turns out that the element does not become an attractor, the normal execution of the behavior function will continue; if on the contrary, the element does become an attractor, then it will gather information from its event window, this information includes all the tiles that are empty, and the tiles that have ADSF elements on them. In order to get the elements as close as possible, the ASDF elements are collected from furthest distance to the smallest one from the central atom whereas the empty tiles are collected from closest distance to furthest from to the central atom. The ordering of these two operations are important because as mentioned before, they allow for the elements stay close. After having these two pieces of data, the element will start allocating the ASDF elements that were found on the outer layers into the inner layers of empty spaces, while also turning this elements into attractors creating some sort of a zombie effect.

The Self-organizing procedure is evaluated based on how close the elements are from the centermost element of the blob. This element is calculated based on the dimension of the bounding rectangle: if the bounding rectangle has widht w and height h, would have its central element in tile $\frac{w}{2}$ and $\frac{h}{2}$[2]. This means that the more similar a blob is to an square, the better.

Section 3.2.2: Localization process

In order to carry out the Localization Process, it is important to have the elements as close together as possible, or at least, as least spread as possible, because this will provide a better sense of self-awareness of each elements position on the blob they belong to.

[2]In case this number is not an integer, it will not be an issue because this is only used as a measurement, we are never really interested on getting the centermost atom in a blob

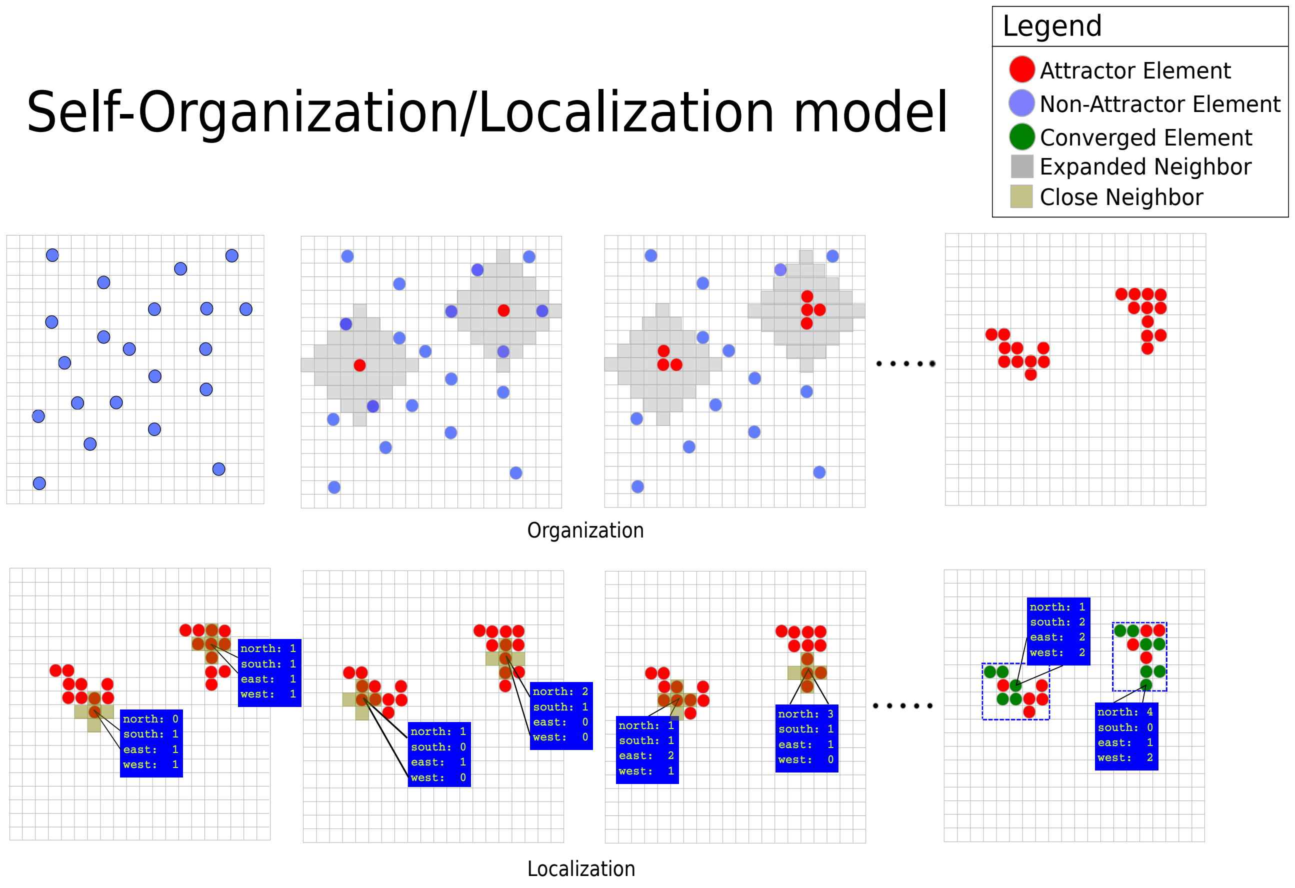

Figure 1: Model of the protocol. The first row specifies the localization procedure in which the elements get together in big blobs. The second row represents the localization procedure, in which the elements start interacting in order to know where they are with respect to the bounding rectangle.

This second stage assumes that, in addition to determining whether an element is an attractor or not, the elements have to have some state stored in them. Each ASDF element has four counters for north, east, south and west which indicate their distance to the boundary rectangle. Algorithm 2 describes the process of localization.

Algorithm 2 Complete the localization process

```
    procedure LOCALIZATION(()$A$)
 2:     if (I have converged?) then
            if (this should check again) then
 4:             if (not convergence?) then
                    SetConvergence(False)
 6:                 AlertNeighbors()
                else
 8:                 this.countN $\leftarrow$ south.countN + 1
                    this.countS $\leftarrow$ north.countS + 1
10:                 this.countE $\leftarrow$ west.countE + 1
                    this.countW $\leftarrow$ east.countW + 1
12:                 if (convergence?) then
                        SetConvergence(True)
14:                 else
                        fixCounts()
```

The algorithm for localization is based on the initial procedure of self-oragnization. In a nutshell, the localization procedure creates a bounding rectangle in every clump and then calculates the distance from every atom in the clump to the boundary element in their own row or column. This then leaves every atom in a state where they know where they are in the whole clump; by knowing how many tiles there are between an atom and the boundary of the clump the atom is in, it is easy to perform operations that take advantage of a bigger neighborhood (Practically all the atoms in your clump).

There are many things we have skimmed over describing the Localization procedure, and because we don't know up to what extent any of the details of our model influence the final result, we will describe all those details in the following paragraphs.

The localization procedure starts right after the elements have clumped down and are all stuck together. The central atom is assument to be a non-converged one, therefore such atom will use their neighbors' counts in order to fix its current counts. Each one of the four immediately Von Neumman Neighbors will be used to update the center atom's counters. For the north position, for example, the center atom will refer to the south atom, and will update its north counter to the south's north counter value plus 1, and this will happen for each one of the other 3 neighbors. After the elements have updated their counters, then they will check for convergence, and in case it wasn't achieved, then they will try to get into an agreement with they neighbors. This procedure of getting in ageement can be viewed as drawing a cross of size 3, for which the central atom is also the central point on this cross, and then for both directions (horizontal and vertical) sets the value for its counter to the highest value of the three atoms. In other words, it is evident that the higher value an atom has for any of their counters, the more knowledge it has about the bounding rectangle. This is justified by the fact that all the counters are initialized to zero, therefore, each time their counters change their values, i.e. the counters get incremented, we can infere that the atoms are getting events and are finally coming to an agreement, or converging.

Before all takes place, an extra check is performed for only those elements that claim to have reached convergence. If an element has reached convergence in a previous iteration[3] and it is still converging, then evertthing seems to be corrent on the localization procedure. If, on the other hand, convergence is not currently achieved, then the element is marked as a non-convergent element, and on the next event, it will behave like any other non-converged atom would do.

We now have described how the process works for giving elements localization knowledge and only the convergence procedure is left to check. There are several way of checking if an element has converged, or even, there are several ways of defining convergence, which is why we will nail down that definition right now, and then it would be easier to describe how to achieve this state. *Convergence* is an state of the MfM machine in which every element have the same counts in a row or in a column, i.e., all the elements know where the bounding rectangle is located, and therefore, they know what their position in the world is. When designing a protocol for reaching convergence it is important to keep in mind robustness, therefore one cannot rely in big neighborhoods or intrincated tricks, the best way is to do it as simple as possible: we opted for checking the immediate vertical and horizontal neighbors and set convergence to TRUE if the values for the counters of the three elements were the same, otherwise convergence has not been reached.

Experiments and Results

We ran several experiments that helped us understand better the way our protocol behaves on different environments. We also designed metrics which can be used to determine how good or bad the two self-organization and localization processes are. Specifically, we implemented our procedure in one new element called the ASDF element, then we did 20 different runs with different amount of elements, starting with 100 elements we incremented the amount of elements by 20 and in every case the experiments were performed in a random fashion. Our results show a high fluctuation in the number of elements, however, on the average, the performance is stable.

[3]An iteration is not really an iteration in the MfM architecture because of the way computation is performed, it is actually an *interaction span*

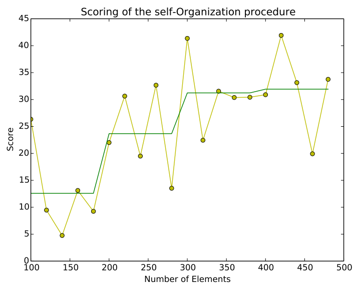

Figure 2: This illustrates the relationship between the number of elements and the overall score of the universe

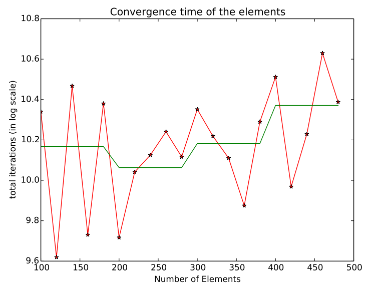

Figure 3: Graph 2. This illustrates the relationship between the number of elements and the total time for convergence for the elements

Self-Organization assessment

The self-organization procedure, as described in the previous section, brings the atoms together in order to start the localization process. It is evident that a good clumping scheme will yield atoms with a center as close as possible from all the atoms. We call this the *score* of a run of the algorithm; once all the atoms are localized, it is easy to calculate the final score: for every element, we know their bounding rectangle coordinates, and therefore we know where the center of the blob is. We then calculate the distance from such atom to the center, and we add this value to the total score. This will give a total value that should be normalized based on the number of blobs and the number of elements. Therefore, the final score of a run is the score obtained by calculating the distances to the central atoms, divided by the number of average atoms per clump.

$$\begin{aligned} \textit{avg-atoms} &= \frac{\text{total atoms}}{\text{number clumps}} \\ \textit{score} &= \frac{\sum_x \text{dist}(x, \text{center})}{\text{avg-atoms}} \end{aligned}$$

Based on the definition of the score, it is easy to see that the lowest this is, the more compact the blob is, therefore, the better. We observed that the score overall fluctuates a lot, however on the average case it stays more or less constant as can be seen in Graph 1. This means there is a direct proportional relation between the number of atoms in the world and the overall score. It was interesting that the best score achieved (4.757) was obtained because in that run, there was only 1 blob formed, hence, the average number of atoms per clump remained high, and the total score was certainly low. This brings up an interesting point: given that our scoring metric dependes on both, the number of atoms on the universe and the number of blobs, it captures the overall picture, rather than every specific atom by themselves. This indidcates then that by increasing the number of blobs, the overall score will remain high, even though the score for the individual blobs might still be low. The fact that the score grows linearly with the number of atoms is a big advantage because it provides solid evidence that the only parameter that affects the score is the number of atoms, hence this scoring is independent of varying the size of the grid, or increasing the total density of the universe: the number of ASDF atoms in the universe will always give a good estimate of what the overall score will be, or in different words, the number of atoms can give us a guarantee on how well we expect to perform on the average case.

Localization assessment

In order to identify how good or bad our protocol performed in the Localization procedure, we chose to evaluate it based on the total convergence time. The convergence time is the total number of iterations it takes for *all* the atoms to get to an state in which they know their position in the clump, i.e. they know how many tiles there are from where they are located to the boundary atoms. We also considered Average Convergence time (AVGCT), however, given that our purpose is that all the atoms should be localized, the AVGCT doesn't really capture this, because there are some pathological cases in which the AVGCT is not really a thruthful measure: for example, in the case in which a big chunk of elements converge on the first few iterations, this elements would bring down the average, but it might have been the case in which the last few elements took a long time to converge, therefore the AVGCT would yield a deceiful measurement.

Figure 3 shows the performance of our protocol based on the number of atoms and the total convergence time. The randomness of our protocol and of the MfM architecture itself show a very fluctuating pattern, however, when the values are averaged (green line) a more visible pattern arises: as in the previous case, the convergence time also has a linear dependency on the amount of atoms, which as explained on the previous subsection, is a good things because it indicates that the convergence time can be predicted based on the number of atoms there are in the world.

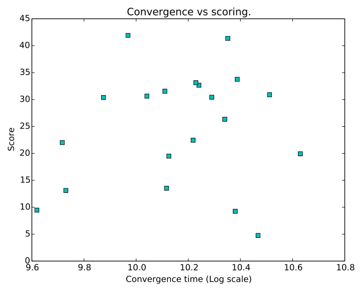

Figure 4: Graph 3. Illustrates the relationship between the convergence time and the overall score of several runs

Conclussions and Future Work

We conclude this article by pointing out that there is a long way to get to a whole model of computation that has complexity even comparable to the complexity of the current computation model, however we also believe that our model should definitely be taken into consideration when performing new tasks for which there is not a clear implementation in the MfM architecture. We have thought about some of the applications that could be performed just by having spacial information of a larger neighborhood, and the posibilities are limitless: just by having every element know where they are on a blob, they can be in charge of some part of the data to be handled, perform some processing on it, and then give it to the bounding atoms, which will then communicate this outputs to the exterior atoms. We could also use this information to create a coordinated movement procedure, that allows each element to decide how many tiles it should move based on the position it occupies in a clump.

Here we also illustrate a way to extend the localization procedure and we provide some arguments on how this will improve the time it takes for the elements to reach a convergence state. The procedure we implemented requires an element to reach the convergence state, based on its neighbors, however there are some pathological cases, as shown in the results section, for which the current procedure takes longer and in an atom or fails to set the correct values for the counters. The proposed procedure for convergence, is based on the fact that convergence can be determined individually for each direction, therefore it is possible to have the north direction of the central atom being correct, however the south direction can still have some discrepancies. Our proposed model, is based on this fact, and then it implements convergence by direction, meaning, an atom is only converged when the four directions have reached the convergence state. We conjecture that this increases the time it takes for an element to converge because some of the information is available sooner, i.e. for the boundary atoms it will take very few iterations to converge in their boundary directions, because those counts are not being modified and the neighbouring elements are also not changing their numbers. We also suggest that by implementing this procedure, it is possible to extend the convergence neighborhood to a larger value (Von Neumman neighborhood of size 2), which will increase the robustness of the system. This options will be explored in future work we perform within the MfM architecture.

References

Ackley, D. H. and Cannon, D. C. (2011). Pursue robust indefinite scalability. In *Proc. HotOS XIII*, Napa Valley, California, USA. USENIX Association.

Silverman, E. and Ikegami, T. (2011). Robustness in artificial life. *International Journal of Bio-Inspired Computation*, 3(3):179–186.

Toffoli, T. and Margolus, N. (1987). *Cellular automata machines: a new environment for modeling*. MIT press.