# Threshold-Based/Market-Based Algorithms Clustering using task-allocation

*Mirco Dotta and Andreas Jaggi*
Swarm Intelligence Course Project
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
{mirco.dotta, andreas.jaggi}@epfl.ch

***Abstract*— In this brief report, we outline the main stages that we have followed in order to build a system able to cluster data of different shape in a self organized fashion. Different theoretical constructs will be analyzed with the purpose to show that it is possible to conceive fast sorting algorithms, even for systems subjected to very low computation abilities.**

KEY WORDS – clustering, sorting, collaboration, swarm intelligence, stigmergy, collective behavior, microscopic model.

## I. Introduction

Gathering data that shares similar properties is a quite common task in todays real life. It is easily possible to find exploitation of this behavior in nature - ants use it to aggregate corpses in a way that looks very similar to a human graveyard - as well as in artificial systems - e.g. collectioning and sorting of heterogeneous data.
These kinds of problems have clearly no optimal computational solution; so they represent an attractive and challenging topic. Many different algorithms have been proposed so far. In general they belong to one of the two well-known families of sorting strategies: hierarchical and partitioner sorting. These clustering methods build their strength on a perfect knowledge of the space of data that needs to be sorted. Whenever this a priori knowledge is not easily available or simply too onerous, then the performance of these centralized strategies tends to decrease considerably.
In this sense the behavior of the *Messor Sancta* ants during their corpse aggregation activity is of particular interest. The observations made on a colony of these ants show that

- The decisions are made in a completely local way, thus each individual decides on its own how to move and how to proceed with a corpse.
- The sum of all the individual decisions has a direct effect on the global structure of the environment. The proceedings of the individual ants make up the creation of clusters of corpses.
- There is no direct communication between the different individuals. Nevertheless ants use pheromone to indirectly communicate spatial informations.

Apparently, a self organized behavior can arise without any active communication between the entities. Information is exchanged through the modification of the environment using different concentrations of pheromone. This phenomenon is called *stigmergy* and is an important concept in the swarm intelligence field.

The lack of global knowledge has a direct and significant impact on the time needed to reach convergence. But this lack also makes the clustering problem very particular since it tries to solve a 'global knowledge' problem without using global knowledge.

Therefore, conceiving a bioinspired, distributed algorithm who is able to solve a data clustering problem in an abstract memory space, is the focus of interest of our project. The ambitious requirements are that the final result is capable to convey the trade-off between, on one hand, the peculiarity of the clustered system, and on the other hand, the complexity and time used to build the specific solution.

## II. Experiments

### A. Framework

An important part of the project has been the development of a software application which allows us to run a clustering simulation. In order to be more flexible and to have a more accurate control over the simulation, we have chosen to produce our own simulation software and not to make use of an already existing one like for example the Webots™environment [**?**]. We modeled our software in a clean object-oriented way to allow easier modification of the different components and to guarantee further extensibility. The source-code is made available under the terms of the GNU Public License in order to make the use and integration of our software in other projects possible without restrictions.

### B. Simulation Set Up

We have two kinds of agents and two kinds of objects which can be put in the environment. The environment consists of a grid of $x$ rows and $y$ columns of cells, without any obstacles but agents, objects and surrounding walls. At the beginning, agents and objects, are uniformly spread on the environment. Each cell can contain at most one element, therefore it is not possible to overlap objects on the same cell. Every agent is selected, at each iteration, one and only one time. When an agent is not carrying an object, he can choose to pickup an object in its neighborhood and go to that cell. When he is carrying an object or does not want to pickup an object (either because he decides so or because their are none available), then he can move to one of the neighbooring cells. When he is carrying an object he has also the possibility to drop that object.

Each agent has the only ability of perfectly knowing the eight, neigbhoring, cells that surround him. They are completely free to move in one of the eight possible directions and, their speed, depend on the loaded object(if they are not charged, they do a movement at each simulation step), so they are more efficient in carrying objects of their same type. We have imposed that, the velocity of an agent that is transporting an object of the same type, corresponds to a movement every two simulation steps; instead, if the agent is loaded with an object of a different type, it can move every three simulation steps. Naturally, every agent is more attracted on carrying object of the same type. This proneness can be modified by tuning the initial collaborative threshold ($\delta^k_{collab}$). Indeed, the greater is the value of this parameter, the more selfish the agents will act(the percentage of agents interested in helping theirs teammates - of a different type - will be very low). At the very beginning, the parameter's value is homogeneous among the agents but, along the simulation, it will be adapted to the modification of the ambient.

### C. Stochastic Model

The system that we have modeled[1] is, almost, completely stochastic. Each time an unloaded agent comes across an object, the probability to pick it up depends mainly on two factors:

1) If the encountered object is of the same type as the agent, then the agent will pick it up every time that, $P^k_{pickup}(i_j)$, is greater or equal to some, fresh, random number $Rand$ (where $Rand \in [0, 1]$).
2) If the encountered object is of a different type than the agent, then the agent will pick it up every time when:

[1]The formulas concerning the probability to pick up an object and to drop it are been taken from [**?**] and slightly adapted to the characteristics of our simulation software

a) the own $\delta^k_{collab}$ value is greater or equal to the $collaborative\_threshold$(set at the beginning of the simulation) and,
b) $P^k_{pickup}(i_j)$ is greater or equal to $Rand$.

The probability $P^k_{pickup}(i_j)$ that an agent of type $k$ picks up an object, say $i$, of type $j$(so $i_j$) is defined as

$$P^k_{pickup}(i_j) = \begin{cases} 0 & \text{if } k \neq j \text{ and } \delta^k_{collab} \leq collab\_thresh, \\ (\frac{K_p}{K_p + f(i_j)})^2 & \text{otherwise.} \end{cases}$$

where $K_p$ is a constant and $f(i_j)$ a local estimation of the density of elements and their similarity to $i_j$.

On the other hand, the probability that an agent will drop a carried element should be in function of the density of similar elements in its surrounding area, and it is important to prevent the creation of mixed clusters and to prevent that two cluster of different types become interconnected. Accordingly, we define the probability to drop an object as

$$P^k_{drop}(i_j) = \begin{cases} 0 & \text{if, in the neighborhood, } \exists\ object_l \text{ s.t. } l \neq j, \\ 2f(i_j) & \text{if } f(i_j) < K_d, \\ 1 & \text{otherwise.} \end{cases}$$

where $K_d$ is a constant. The density function $f(i_j)$ (used in the definition of $P^k_{pickup}$ and $P^k_{drop}$) is defined as

$$f(i_j) = \begin{cases} \frac{1}{|neighborhood|}(\sum_n^{neighborhood} 1 - diss(i_j, n)) & \text{if } f > 0, \\ 0 & \text{otherwise.} \end{cases}$$

where $neighborhood$ represents the set containing the neighbor cells of agent $k$. The dissimilarity $diss(i_j, n)$ among an object $i$ of type $j$ and each neighboring cell $n$ is computed as

$$diss(i_j, n) = \begin{cases} 0 & j = type(n), \\ 1 & \text{otherwise.} \end{cases}$$

$type(n)$ is a function that, given a cell as parameter, it will return the type of the element that is holding the cell or $EMPTY$ if the cell is free.

Finally, we still need to define how the collaboration parameter $\delta^k_{collab}$ of each agent $k$ will change along the simulation

$$\delta^k_{collab} = \begin{cases} \delta^k_{collab} + \frac{N_j}{3} & \text{if } N_j > 0 \\ 1 & \text{if } \delta^k_{collab} < 1 \\ \delta^k_{collab} - \frac{Rand}{8} & \text{otherwise.} \end{cases}$$

where $N_j$ is the number of neighbooring cells that contain an object of type $j$ such that $j \neq k$ (where $k$ is the type of the agent). Furthermore, from the definition of $\delta^k_{collab}$, the threshold has to be greater or equal to one(moreover, even if we use a smaller value, the threshold will be set to one).
Considering that the agents in our systems do not have any particular skills (but the capacity to perceive the eight surrounding cells and to keep the current direction), it was difficult to bring to light a rule that was able to quickly adapt even to the smaller change in the environment. Despite its raw nature, the formula that we have proposed is able to handle different situations in a fair manner.

Everything we have discussed so far are just block of the system we have built. Still, we needed to settle the formulas in order to describe how the agents act at the microscopic level.

### D. Microscopic Model

Using the formulas established in the stochastic model we can create a relatively concise microscopic model of the agents behavior. Fig.**??** shows a probabilistic finite state machine (PFSM) whose state-to-state transitions depend on the interaction among an agent and the surrounding environment. Initially agents and objects are uniformly spread in the environment, but throughout the simulation this assumption does not hold anymore since this is exactly the opposite of the goal the clustering process. To formalise our microscopic model, we use the hypothesis that objects and agents maintain an uniform distribution over the time. Therefore this simplification will likely produce some variance between the expected and the simulation results.
The probability of encountering an agent can be computed as $p_a = \frac{A_a}{A_e}$, where $A_e$ is the area of the environment and $A_a$ is the total area take up by the agents. Similarly, $p^j_o(t) = \frac{A^j_o(t)}{A_e}$ represents the probability of finding an object $j$. As agents can carry objects, $A_o(t)$ tends to vary at each step of the simulation, so $A^j_o(t)$ will represent the objects of type $j$ that are on the ground at time $t$. Finally, the total probability of encountering any other agent on the arena can be computed as $p_A = 8(N_0 - 1)p_a$, where $N_0$ is the total number of agents in the arena. Thus, using Fig.**??** we're able to describe the probability that, an unloaded agent $k$ will pick up an object $i_j$ at time $t$, as $p^k_{pick}(t) = 8P^k_{pickup}(i_j)p^j_o(t)$. Similarly, $p^k_{drop}(t) = 8P^k_{drop}(i_j)p^j_o(t)$ represents the probability that a loaded agent $k$ will drop an object $i_j$ at time $t$. The careful reader could be wondering about the factor 8 present in the $p^k_{pick}(t)$ and $p^k_{drop}(t)$ formulas just exposed, this factor is due to the setup of our simulation software, where the encountering of an object through an agent is considered to be the case only when the object is located in one of the eight cells neighbooring the agent. The probability that there is an object in this neighborhood (e.g. the probability of an agent to encounter an object) is therefore eight times the probability that an object is on a single cell. This affects both the $p^k_{pick}(t)$ and the $p^k_{drop}(t)$ probabilities since an object can only be picked up when it is encountered and an object can only be dropped near a cluster which is only the case when another object is encountered.

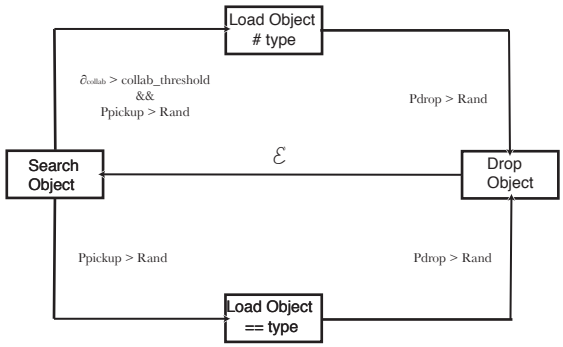


Fig. 1. Microscopic Level: PFSM

### E. From Microscopic to Macroscopic Level

Roughly speaking, an agent, at any time, is searching for objects. May it be for picking it up when he is not carrying another object, or otherwise for dropping the carried object to form a cluster with the just discovered one. The macroscopic behavior of the swarm will, therefore, directly depend on the algorithm implemented to control the movement of the agents. During our project we have followed two

different paths to understand, in which way, the global final solution could be substantially affected by a different microscopic decision on the deplacement.
The first strategy was simply to take, at each step, a random walk[2] in the environment. So the decision on the next cell where to move was completely casual. The weakness of this solution was evident, the agents were wasting a lot of time wandering around a considerable small portion of the environment. Consequently, we needed a very long time to achieve a good sorting(unless the environment had a small size).
The second idea[3] come out after one swarm intelligence lecture. We have been fascinated by the *Puck Clustering* experiment[**?**], so the decision to implement a similar version. The idea is basically to move 'forward' unless we face some obstacle (e.g. a wall, an object or another agent). In this way, the decision directly depend on 1) in which state we are and 2) the previous direction(besides, this perfectly describe a discrete Markov Chain). Fig.**??** should clarify the concept.

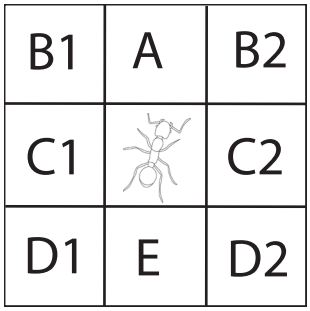


Fig. 2. Straight strategy: An agent (represented as an ant) coming from $E$, first tries to go to $A$ (when the cell is free). If this is not possible he checks wether $B1$ or $B2$ are free, and tries to go there. If both are free, the choice is made in a equally probabilistic manner, since preferring $B1$ or $B2$ would leads to a biased movement which causes all agents to move to the same corner. If it is not possible to go to $B1$ or $B2$, the agent checks wether $D1$ or $D2$ is free and tries to go there (also here the decision is made in the same equally probabilistic manner as before for the same reasons). If $D1$ and $D2$ are not free, the he checks wether $C1$ or $C2$ are free and tries to go there (with the same equally probabilistic decision algorithm as before). If the agent still has not found a free cell, there rests no other choice than to go back to $E$.

After studying Fig.**??** you are surely wondering why we try to go to the $D$ cells instead of trying the $C$ cells. This is to prevent the agents from following the walls to often. Since our simulation environment is rectangular and the cells are positioned in a regular grid, following the walls to often would led to a situation where almost all agents follow the walls and only a few try to do clustering of objects in the middle of the area. So we have changed the strategy such that an agent, when he runs into a wall, first considers going in a diagonal backward direction and only considers following the wall when this is not possible. This change immediately resolved the wall-following issue.

Accordingly to our expectation, the collective behavior with the *straight strategy* was considerably improved and a convergence is observable also for big environments with a lot of uniformly distributed objects. Section **??** compares the impacts on convergence and performance in more depth.

### F. Evaluation of the performance

In a clustering simulation it is always difficult to describe whether a given situation is good or bad. We basically wanted to exploit a tradeoff between the accuracy of the final solution on one hand and, on the other hand, the increasing difficulty to obtain a single cluster with the growth of the objects present in the system. Starting from these needs, we first try to formalise these two factors.

- **Accuracy** The accuracy of the final solution can be computed as
$$\frac{1}{clusts^k}(\frac{r^k_{min}}{d^k_{avg}})$$
where $clusts^k$ and $r^k_{min}$ represents, respectively, the number of clusters of objects $k$ and the minimal radius of the best solution(so only one cluster), that we can obtain starting from $nb\_objs^k$. Finally,$d^k_{avg}$ is the average distance between every object of type $k$ on the ground and an abstract center of mass existing among the objects. Fig. **??** should help to clarify the concepts.
- **Toughness** The difficulty of finding the optimal solution is in relation with the initial amount of objects $k$ spread in the environment
$$\frac{1}{clusts^k}(\log\frac{nb\_objs^k}{clusts^k})$$
where $clusts^k$ and $nb\_objs^k$ represent, respectively, the number of clusters and the initial number of objects (on the ground) of type $k$ in the arena.

Accordingly, the final fitness function $f^k$ is nothing more than the sum of the two formulas that we have developed above

$$f^k = min(\frac{1}{clusts^k}(\frac{r^k_{min}}{d^k_{avg}}), 0.5)+min(\frac{1}{clusts^k}(\log\frac{nb\_objs^k}{clusts^k}), 0.5)$$

The introduction of the $min$ function is necessary to keep the maximal value obtainable less or equal to one.

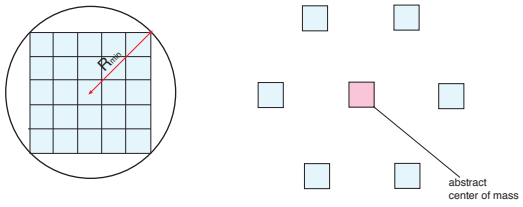


Fig. 3. (Left) Given a certain initial amount of objects of some type $k$ we find the minimal radius $r^k_{min}$(in the figure $R_{min}$) of the best cluster formation. (Right) The abstract center of mass among the object used to compute $d^k_{avg}$ that represents the average distance of all objects from the centroid(center of mass)

## III. Results

### A. Experiments

In order to validate the theoretical framework provided in the previous section, we have made a series of experiences. The basic settings we have used along the tests are the following:
An environment grid of 50 rows and 50 columns, 20 agents of each type (the two different types will be referred, from now on, as $A$ and $B$), 100 objects of each type and, the value of the collaborative threshold, set to 2.0. The default movement algorithm used in the experiments is the *straight strategy*, unless explicitly stated otherwise. For each run we have done 10'000 iterations. In order to obtain more accurate values which are not influenced by occasionally occurring

[2]we will reference to it as *random walk strategy*
[3]we will reference to it as *straight strategy*

phenomenon, each experience was launched 10 times and (except if stated otherwise) the average values were used.
In the first experience we used the default settings described above but with changing the movement algorithms (e.g. 10 runs with the straight strategy and 10 runs with the *random walk strategy*). Then, for the experience two, we have changed the proportion between agents $A$ and $B$ (using 30 agents of type $A$ and only 10 of type $B$). For the last experience, we have changed the proportion of objects in the arena, using 130 objects of type $A$ and only 70 of type $B$ while maintaining the remaining parameters according to the default case.

The first experience was made mainly to analyze the differences between the *random walk strategy* and the *straight strategy*. This in order to prove our intention that the *straight strategy* performs significantly better than the *random walk strategy*. But it was also made to have a common base to compare the results of the other experiences.

The goal of the experiences two and three, is to verify that agents can adapt their behavior according to the necessity they face in the environment. So, even if we have two different castes of agents, there is a certain level of flexibility among the colony that allows to obtain comparable results to the default case. This flexibility can be tuned by modifying the value of $\delta^k_{collab}$, the collaborative threshold of the agents. In our experiences we have used a $\delta^k_{collab}$ set to two, which showed to be adequate after performing numerous tests. But still, we ca not be sure that this is the optimal value that we could ever choose. If we would really have wanted to find the optimal value, then we could have done a PSO (or GA) based search to find it.

Fig.**??** gives a graphical summary of the obtained fitness results. Additionally, we provide a table containing, for each experience, the obtained fitness means and standard deviation $\theta^2$.

| Experience | Fit A | $\theta^2_A$ | Fit B | $\theta^2_B$ |
|---|---|---|---|---|
| 1 | 0.69 | 0.05 | 0.72 | 0.1 |
| 1 (random walk) | 0.51 | 0.03 | 0.47 | 0.06 |
| 2 | 0.73 | 0.07 | 0.81 | 0.13 |
| 3 | 0.74 | 0.11 | 0.74 | 0.14 |

There are some consideration that are worth to be reported. First there is a certain balance among the obtained fitness, this is a good property of the system (we will deepen this aspect in the next paragraph). Secondly, we can appreciate that the *straight strategy* performs considerably better than the random one (the fitness quality of the solution has increased $\sim 150\%$). Finally, it is important to analyse the standard deviation. As a matter of fact, the values yield to consider that the system is subjected to fluctuation. This it is not astonishing since we are dealing with a stochastic system.

### B. Convergence / Comparison of the movement strategies

An important property that any clustering algorithm should, at least, provide is that, eventually, it converges. Naturally, the smaller is the time needed to obtain convergence (and find an optimal solution), the better is the sorting strategy. In our model, the *straight strategy* is able to converge, statistically, after $\simeq 20'000$ iterations. In Fig.**??** are shown the different stages of a run with the *straight strategy* including the final convergence.

We can appreciate that after a quite small number of iterations($\sim 5'000$) we already have a well sorted system (bottom left of Fig.**??**). Starting from this point, the performance begins to suffer and the convergence is met only after a big number of steps. There are several explications to this phenomena. Mainly, the fact that the system is probabilistic and the agents' skills are narrowed to the bare minimum, imply that the destruction of a small cluster, in favor of a bigger one is, basically, a lucky event (e.g there is no part in the algorithm

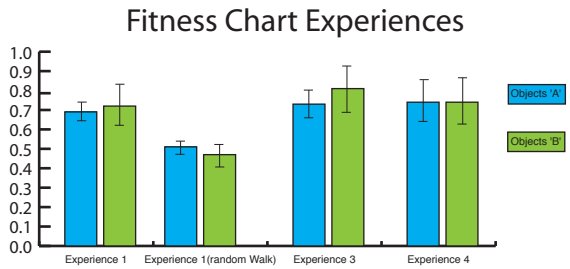


Fig. 4. The default values used for the experiences are: A grid 50x50, 20 agents of each type, 100 objects of each type and the *straight strategy* unless stated otherwise. In experience two we have changed the proportion of agents to 30 agents of type $A$ and only 10 of type $B$. Then, for the last one, we have changed the proportion of objects to 130 objects of type $A$ and 70 of type $B$ (respect to the default case). The lines cutting the bars represent the standard deviation $\theta^2_j$ obtained for each test.

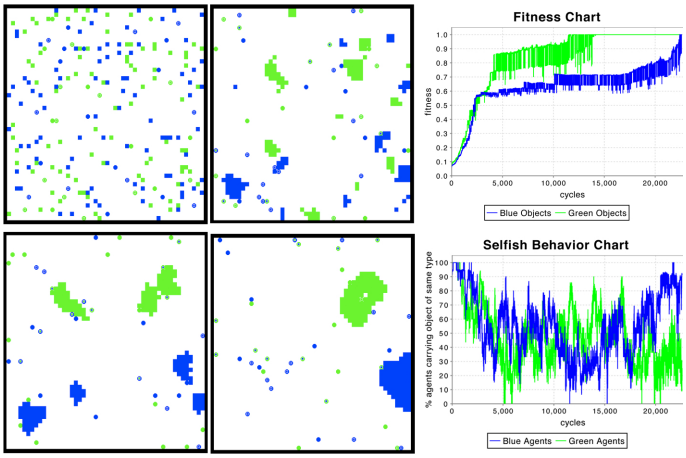


Fig. 5. (Top Left) Initial Situation. (Top Middle) After $2'000$ iterations. (Bottom Left) After $5'000$ iterations. (Bottom Middle) Convergence, after $\sim 22'000$ iterations. (Top Right) Fitness chart during the simulation. (Bottom Right) Percentage of agents that are carrying an object of the same type as they are. This also gives an idea on the amount of collaboration the agents are doing.

controlling the agents' behavior that explicitly tries to destroy an existing small cluster, but still it eventually occurs). It would have been interesting to enhance the capabilities of the agents with, for example, stigmergy based skills like those which can be observed in ant colonies with the pheromone the ants use to mark the ways to the food sources. With such an ability to mark the environment in which the agents are picking up the objects, the behavior of the entire swarm could become much more coordinated and we can presume that a significantly faster convergence could be observed.

Comparing the two algorithms that we have implemented for the movement strategies shows that the *straight strategy* reaches impressive performance compared to the *random walk strategy*. This is most impressively proven by looking at the number of iterations needed for total convergence (only 1 cluster for each object type). With the *straight strategy* convergence is normally obtained after about $20'000$ iterations whereas the *random walk strategy* needs more than $100'000$ iterations to fully converge. So we can say that the *straight strategy* converges 5 times faster than the *random walk strategy*. This is also clearly visible when you observe the environment during the experiences. At $20'000$ iterations the *straight strategy* has normally converged and the environment looks like the

one shown in the bottom middle image of Fig.**??**. But with the *random walk strategy* at $20'000$ iterations the environment looks like the top middle image of Fig.**??** or like the bottom left image of Fig.**??**.

But as the overall agent controlling algorithm is not changed, the fitness charts of two experience with the same parameters but with different movement strategies look very similar. The fitness chart of the *random walk strategy* is 'just' a stretched version of the fitness chart of the *straight strategy*.

## IV. Conclusion

First we can say that the formulas established by Lumer,Faieta provide a working clustering process even if engaged in a significantly different setup. This is especially positive since our simulation setup is quite different from the one in Lumer,Faieta (we use a bigger environment with completely uniformly distributed objects, whereas in Lumer,Faieta[**?**] the environment is smaller and the objects are initially spread following a gaussian distribution with four areas of collection, therefore forming a 'pre-clustered' environment) and also because we are using the basic version of their algorithm (the agents don't have a memory or history) with our enhancement consisting of a variable collaboration threshold (which also does not use history).

Also since our environment is quite different, the basic algorithm needs a very big number of iterations to converge to the best possible solution (even with the collaboration threshold enhancement). There are several explications for this phenomena.

First of all, the very limitated ability of the agents. In fact, every decision they made is in function of the neighboring surrounding environment they perceive. Nonetheless, they are able to adapt their attitude along the simulation and a collective behavior is observable. Indeed, this faculty is of much more interest in systems submitted to very low computational ability. Additionally, from the observations on the simulations that we have done, we can perceive that, after having reached a first raw sorted situation, agents happen to waste their time in picking and dropping the objects on the same cluster. Furthermore, often they are able to destroy a cluster when, in the same iteration step, different agents pick up an object from the same cluster. So, we can state that the destruction of a cluster is more likely to happen if there is a certain level of cooperation among the agents (this remembers us the pull stick experiment made by [**?**] in which the successful conclusion of the swarm tasks require that two robots collaborate to solve the local problem they face).

Another source for causing the convergence only after a very big number of iterations is the movement algorithm who steers the agents. Since in our implementation an agent can only move by one cell at each iteration (and the speed at which is allowed to move depend on whether it is unloaded or loaded) iteration ) and since the environments used in our simulations are rather big, it becomes obvious that the algorithms steering the agents movement are crucial for the convergence time of the clustering process. This becomes clearly visible when comparing the convergence times of the *random walk strategy* and the *straight strategy* which differ by a factor of $5$. So when generally using our algorithm in big environments with randomly distributed objects, our *straight strategy* movement algorithm performs much better since he focuses more on 'non-local' movement than on 'local' movement.

Finally, we have tried, all along the project, to supply the agents with more sophisticated features (like a memory of the last dropped object - in order to avoid to take the same object twice - or an adaptive timeout on the carried object in order to avoid that an agent wastes too much time to find a cluster where he can drop the loaded object). In any of these cases, the performances after the modifications have considerably decreased. This fact, let us thinking that exasperate the microlevel of an agent with complicate skills does not yield to a better behavior of the swarm.

### A. Acknowledgments

We would like to thanks Christopher Cianci for having always been at disposal and helped all along the project with insightful ideas and, when necessary, some critics. Thank you.

*Make everything as simple as possible, but not simpler.*
*Isaac Newton*

## References

[Lumer,Faitea] E.D. Lumer and B. Faitea, *Diversity and Adaptation in Populations of Clustering Ants*.

[Martinoli,Agassounon] A. Martinoli and K. E.W. Agassounon, *Modelling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation*.

[Webots] `http://www.cyberbotics.com/`

[Beckers,Holland,Deneubourg] R. Beckers, O.E. Holland and J.L. Deneubourg, *From Local Actions to Global tasks: Stigmergy and Collective Robotics*, 1994