A multidimensional flocking algorithm for clustering spatial data

Antonio Augimeri, Gianluigi Folino, Agostino Forestiero and Giandomenico Spezzano
Institute for High Performance Computing and Networking (ICAR-CNR)
augimeriantonio@email.it, (folino, forestiero, spezzano)@icar.cnr.it

***Abstract*—In this paper, we describe the efficient implementation of M-Sparrow, an adaptive flocking algorithm based on the biology-inspired paradigm of a flock of birds. We extended the classical flock model of Reynolds with two new characteristics: the movement in a multi-dimensional space and different kinds of birds. The birds, in this context, are used to discovery point having some desired characteristics in a multidimensional space. A critical point of the algorithm is the efficient search of the k-neighbors in a multidimensional space. This search was efficiently implemented using the ANN libraries.**

I. Introduction

Ants'colonies, flocks of birds, termites, swarms of bees etc. are agent-based insect models that exhibit a collective intelligent behavior (*swarm intelligence*) [2] that may be used to define new **distributed clustering algorithms**. In these models, the emergent collective behavior is the outcome of a process of self-organization, in which insects are engaged through their repeated actions and interaction with their evolving environment. Intelligent behavior frequently arises through indirect communication between the agents using the principle of **stigmergy** [6]. This mechanism is a powerful principle of cooperation in insect societies. According to this principle an agent deposits something in the environment that makes no direct contribution to the task being undertaken but it is used to influence the subsequent behavior that is task related. Swarm intelligence (SI) models have many features in common with Evolutionary Algorithms (EA). Like EA, SI models are population-based. The system is initialized with a population of individuals (i.e., potential solutions). These individuals are then manipulated over many iteration steps by mimicking the social behavior of insects or animals, in an effort to find the optima in the problem space. Unlike EAs, SI models do not explicitly use evolutionary operators such as crossover and mutation. A potential solution simply 'flies' through the search space by modifying itself according to its past experience and its relationship with other individuals in the population and the environment.

These algorithms show a high level of robustness to change by allowing the solution to dynamically adapt itself to global changes by letting the agents self-adapt to the associated local changes.

In this paper, we present a prototype using a new algorithm based on the concepts of a flock of birds that move together in a complex manner with simple local rules, to explore multidimensional spaces for searching interesting objects. The algorithm is an extension of the classical flock model of Reynolds with two new characteristics: the movement in a multi-dimensional space and different kinds of birds. The birds, in this context, are used to discovery point having some desired characteristics in a multidimensional space. The implementation is based on SWARM [7], a software package for multi-agent simulation of complex systems, developed at the Santa Fe Institute.

From an efficiency point of view the most critical point of the algorithm is the efficient search of the k-neighbors (i.e. the cardinality of the neighborhood) in a multidimensional space. This search was efficiently implemented using the ANN (Approximate Nearest Neighbor) libraries.

The remainder of this paper is organized as follows. Section 2 describes the adaptive flocking algorithm, section 3 shows how the approach was applied to multidimensional spaces. Section 4 shows some interesting experimental results about the efficiency of the algorithm and its efficacy in finding interesting patterns. Finally section 5 draws some conclusions.

II. The adaptive flocking algorithm

The classical flocking model, introduced by Reynolds [8], moves in a two dimensional space and all the birds have the same characteristics. In the next subsections, we describe the two extensions introduced in M-Sparrow, the use of birds having different characteristics (represented by different colors) and the movement in a multidimensional space.

A. Reynolds' original flock model

The flocking algorithm was originally proposed by Reynolds as a method for mimicking the flocking behavior of birds on a computer both for animation and as a way to study emergent behavior. Flocking is an example of emergent collective behavior: there is no leader, i.e., no global control. Flocking behavior emerges from the local interactions. Each agent has direct access to the geometric description of the whole scene, but reacts only to flock mates within a certain small radius. The basic flocking model consists of three simple steering behaviors: separation, cohesion and alignment.

Separation gives an agent the ability to maintain a certain distance from others nearby. This prevents agents from crowding too closely together, allowing them to scan a wider area. Cohesion gives an agent the ability to cohere (approach and form a group) with other nearby agents. Steering for cohesion can be computed by finding all agents

in the local neighborhood and computing the average position of the nearby agents. The steering force is then applied in the direction of that average position. Alignment gives an agent the ability to align with other nearby characters. Steering for alignment can be computed by finding all agents in the local neighborhood and averaging together the 'heading' vectors of the nearby agents.

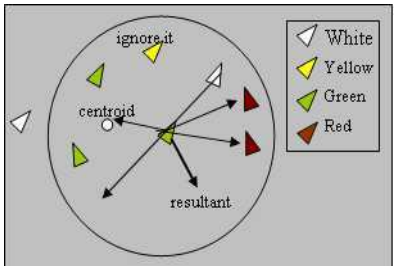


Fig. 1. Computing the direction of a green agent.

B. An adaptive colored flocking algorithm

M-SPARROW extends the Reynolds' flocking algorithm, described in the previous subsection, considering four different kinds of agents, classified on the basis of some properties of data in their neighborhood. These different kinds are characterized by a different color: red, revealing interesting patterns in the data, green, a medium one, yellow, a low one, and white, indicating a total absence of patterns. In practise, the flock follows an exploring behavior in which individual members (agents) to first explore the environment searching for goals whose positions are not known *a priori*, and then, after the goals are located, all the flock members should move towards these goals. Agents search the goals in parallel and signal the presence or the lack of significant patterns into the data to other flock members, by changing color. The main idea behind our approach is to take advantage of the colored agent in order to explore more accurately the most interesting regions (signaled by the red agents) and avoid the ones without clusters (signaled by the white agents). Red and white agents stop moving in order to signal this type of regions to the others, while green and yellow ones fly to find more dense clusters. Indeed, each flying agent computes its heading by taking the weighted average of alignment, separation and cohesion (as illustrated in figure 1). The entire flock then moves towards the agents (*attractors*) that have discovered interesting regions to help them, avoiding the uninteresting areas that are instead marked as obstacles. The color is assigned to the agents by a function associated with the data analyzed. In practice, the agent computes the property of the explored point and then it chooses the color (and the speed) in accordance to the simple rules showed in table I.

So *red*, reveals a high density of interesting patterns in the data, *green*, a medium one, *yellow*, a low one, and white, indicates a total absence of patterns. The color is used as a communication mechanism among flock members to indicate them the roadmap to follow. The roadmap is adaptively adjusted as the agents change their color moving to explore data until they reach the goal.

$$\begin{array}{ccccc} property > threshold & \Rightarrow & mycolor = red & (speed = 0) \\ \frac{threshold}{4} < property \leq threshold & \Rightarrow & mycolor = green & (speed = 1) \\ 0 < property \leq \frac{threshold}{4} & \Rightarrow & mycolor = yellow & (speed = 2) \\ property = 0 & \Rightarrow & mycolor = white & (speed = 0) \end{array}$$

TABLE I
ASSIGNING SPEED AND COLOR TO THE AGENTS

Green and yellow agents compute their movement observing the positions of all other agents that are at most at some fixed distance (*dist_max*) from them and applying the rules of Reynolds' [8] with the following modifications:

- *Alignment* and *cohesion* do not consider yellow agents, since they move in a not very attractive zone.
- *Cohesion* is the resultant of the heading towards the average position of the green flockmates (centroid), of the attraction towards red agents, and of the repulsion by white agents.
- A *separation* distance is maintained from all the agents, whatever their color is.

Agents will move towards the computed destination with a speed depending from their color: green agents will move more slowly than yellow agents since they will explore denser zones of clusters. An agent will speed up to leave an empty or uninteresting region whereas it will slow down to investigate an interesting region more carefully. The variable speed introduces an adaptive behavior in the algorithm. In fact, agents adapt their movement and change their behavior (speed) on the basis of their previous experience represented from the red and white agents.

```
for i=1 ... MaxIterations
        foreach agent (yellow, green)
                age=age+1;
                if (age > Max_Life)
                        generate_new_agent();die();
                endif
                if (not visited (current_point))
                        property = compute_property(current_point);
                        mycolor= color_agent(property);
                endif
        end foreach
        foreach agent (yellow, green)
                dir= compute_dir();
        end foreach
        foreach agent (all)
                switch (mycolor){
                    case yellow, green: move(dir, speed(mycolor)); break;
                    case white: stop(); generate_new_agent(); break;
                    case red: stop(); generate_new_close_agent(); break; }
        end foreach
end for
```

Fig. 2. The pseudo-code of M-SPARROW.

During simulations a cage effect, was observed; in fact, some agents could remain trapped inside regions sur-

rounded by red or white agents and would have no way to go out, wasting useful resources for the exploration. So, a limit on their life was imposed to avoid this effect; hence, when their age exceeded a determined value (maxLife) they were killed and were regenerated in a new randomly chosen position of the space.

C. Exploring multidimensional spaces

We wanted use our adaptive flocking algorithm in order to explore multidimensional space for searching point having desired properties. A continuous data point can be represented in a multidimensional Euclidean space, simply normalizing its attributes. Our algorithm can search for any kind of properties. In particular, we describe a useful property for the task of clustering. Given a radius (*Eps*) and a minimum number (*MinPts*) of points. A core point is a point with at least MinPts number of points in an Eps-neighborhood of the itself. Searching points having these characteristic could be useful for different task (i.e. db-scan [3] use these points to perform the task of clustering databases). In the experimental section, we will show the evaluation of our algorithm in effectively cope with this task.

In the following, we give a more formal description of the extension of the flocking algorithm to the multidimensional space. Consider a multidimensional space with dimension d. Each bird k can be represented as a point in this space, having coordinates $x_{k1}, x_{k2}, \ldots, x_{kd}$ and having direction $\theta_{k1}, \theta_{k2}, \ldots, \theta_{kd}$, where θ_{ki} represent the angle between the new direction of the bird k (computed using the rules of the previous subsection) and the axis i. Each bird moves following the the rules of the previous subsection having speed v_k. Then, for each iteration t, the new position of the bird k can be computed as:

$$\forall i = 1 \ldots d \qquad x_{ki}(t+1) = x_{ki}(t) + v_k \times c_{ki} \quad (1)$$

where c_{ki} represents the projection along the i axis of the direction of the boid k. Note that each components is obtained summing the respective three components of alignment, separation and cohesion (i.e. $c_{ki} = c\_alignment_{ki} + c\_separation_{ki} + c\_cohesion_{ki}$). In a multidimensional space, we can compute the components as:

$$\begin{aligned} c_{k1} &= \prod_{j=1}^{d-1} cos(\theta_{kj}) \\ c_{ki} &= sin(\theta_{ki-1}) \prod_{j=i}^{d-1} cos(\theta_{kj}) \qquad i = 2 \ldots d \end{aligned} \quad (2)$$

these formulas can be computed as a generalization of the three-dimensional case illustrated in figure 3.

From a computational point of view, a critical point of our algorithm is the efficient search of the k-neighbors in a multidimensional space. Computing exact nearest neighbors can be very expensive when dimension increases.

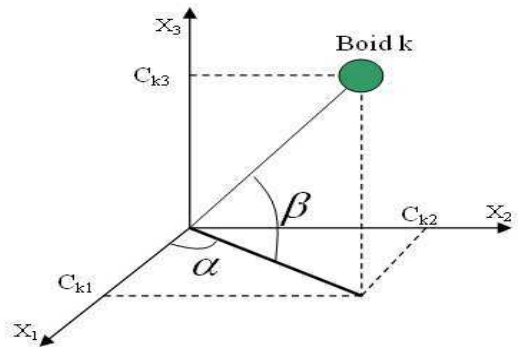


Fig. 3. Computing the components of the boid in a three-dimensional space.

ANN (Approximate Nearest Neighbor) [1] is a library written in C++, which supports data structures and algorithms for both exact and approximate nearest neighbor searching in high dimensional spaces. As showed in figure 4, we integrated ANN libraries using Java Native Interface with M-Sparrow in order to efficiently compute the neighbors necessary to our algorithm.

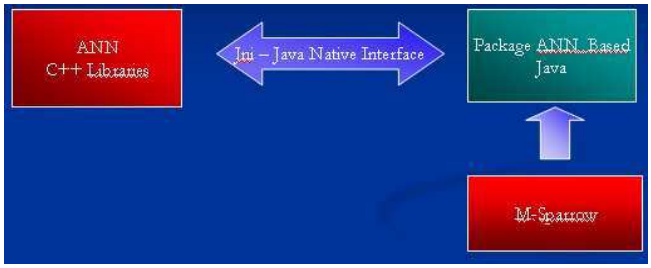


Fig. 4. Integrating M-Sparrow and ANN libraries.

III. Experimental results

In two previous papers, we demonstrated the goodness of our algorithm in discovering clusters with different sizes, shapes in noise data [5] and also with different densities [4] in a two dimensional space.

Now, we want to show how the algorithm works in a multidimensional space. We have built two dataset, each one constituted by two gaussian distributions with different densities of 1500 points. The first was three dimensional and the latter four dimensional. We run our algorithm using 200 birds for 600 iterations with k = 50 and radius = 20. It succeeds in separating almost perfectly the two gaussian distributions in both the cases. The three dimensional case is illustrated in figure 5.

Furthermore, we want to verify the efficiency of the ANN based implementation and then we run our algorithm comparing execution times of the latter ANN-based version with the previous that used brute force computation for searching the k-neighbors. The results of this comparison for a three dimensional case are reported in figures 6 a and b. Experiments show that the ANN libraries outperforms the brute force approach in all the cases and the differ-

ence is really considerable when the number of neighbors to search is greater than 50. If we consider datasets with dimension larger than 3, ANN outperforms the brute force approach by at least two order of magnitude, also for small values of k.

IV. Conclusions

We have presented an efficient implementation of an adaptive flocking algorithm that efficiently search multidimensional spaces. The implementation is based on the ANN libraries performing an efficient search of the k neighbors. Experiments showed that the algorithm is able to separate clusters in multidimensional spaces and it outperforms the previous brute force approach in terms of execution time.

References

[1] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of ACM*, 45(6):891–923, 1998.

[2] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Swarm intelligence: From natural to artificial systems. *J. Artificial Societies and Social Simulation*, 4(1), 2001.

[3] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

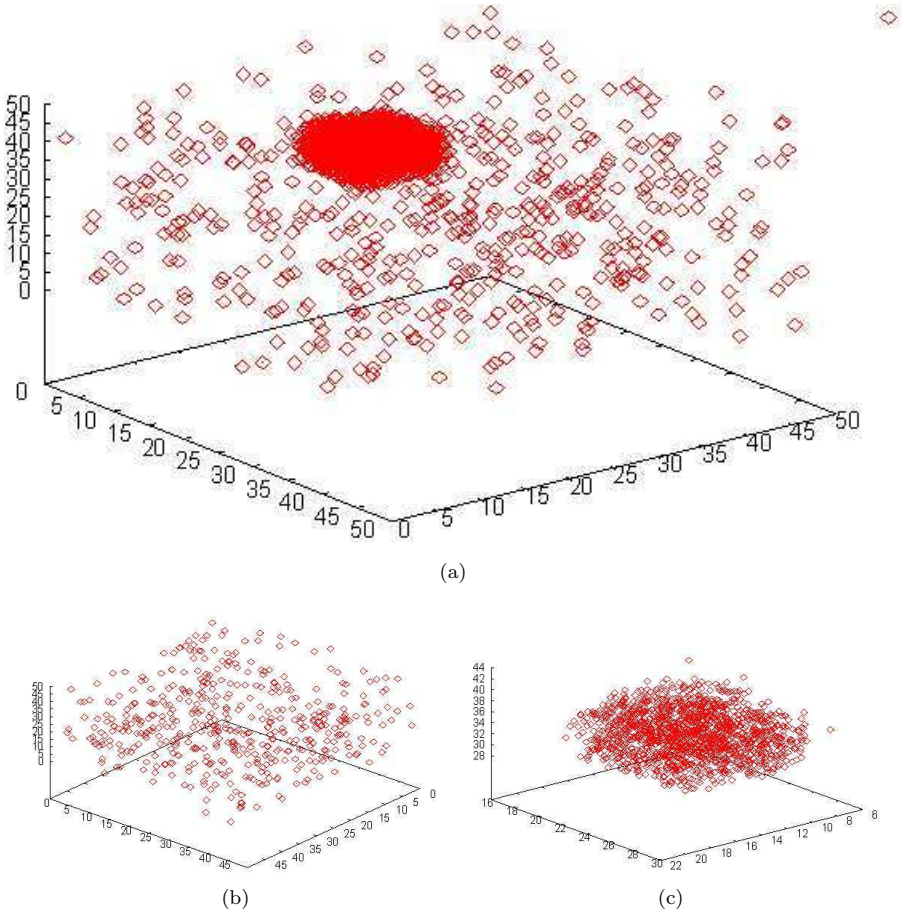
[4] Gianluigi Folino, Agostino Forestiero, and Giandomenico Spezzano. Swarming agents for discovering clusters in spatial data. *ispdc*, 2003.

[5] Gianluigi Folino and Giandomenico Spezzano. An adaptive flocking algorithm for spatial clustering. In *PPSN*, pages 924–933, 2002.

[6] P.P. Grass. *La Reconstruction du nid et les Coordinations Inter-Individuelles chez Beellicositermes Natalensis et Cubitermes sp. La Thorie de la Stigmergie : Essai d'interprtation du Comportement des Termites Constructeurs in Insect. Soc. 6.* Morgan Kaufmann, 1959.

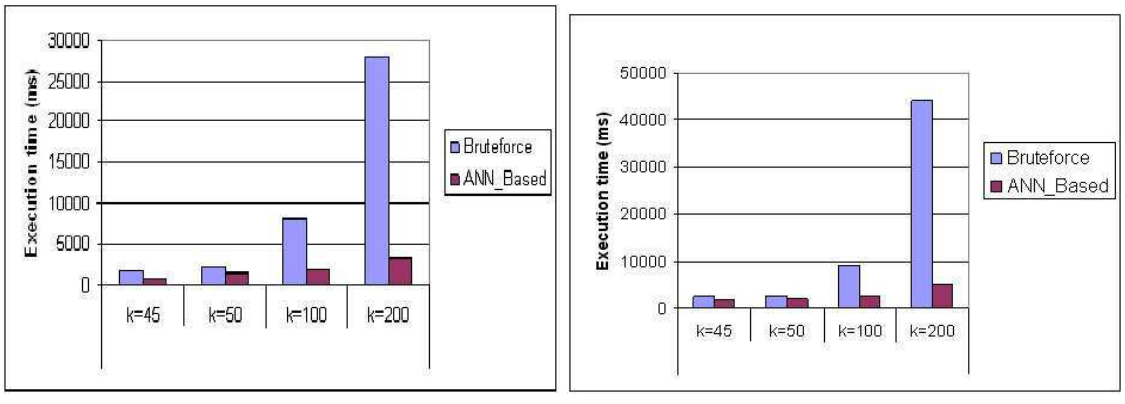
[7] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system, a toolkit for building multi-agent simulations, 1996.

[8] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM Press.



(a) (b) (c)

Fig. 5. Gaussian dataset. a) Original Gaussian dataset used in the experiments; b) First cluster extracted by the algorithm; c) Second cluster extracted by the algorithm.



(a) (b)

Fig. 6. Execution time of the searching phase for the ANN-based version vs the Brute-force approach: a)Radius = 20; b) radius = 50.