University of Ljubljana
Faculty of Computer and Information Science

Reimplementing the FRIsheeping herding algorithm using fuzzy logic

Veljko Dudić, Noah Novšak, Petra Kuralt, and Timotej Košir

Collective behavior course research seminar report

November 19, 2023

Iztok Lebar Bajec | professor | mentor

Concisely summarize the research objectives, methodology, key findings, and implications.

fuzzy logic | herding algorithm | genetic algorithm

This study explores the integration of fuzzy logic into herding algorithms as an innovative approach to enhance adaptability and nuanced decision-making within simulated group dynamics. Initially, we build upon the Strömbom algorithm, implemented in Unity environment, introducing fuzzy logic to refine the behavior of entities within the herd. Subsequently, our investigation extends to the implementation of herding mechanism using genetic algorithms. This utilization of fuzzy logic and genetic algorithms aims to create a comprehensive and robust framework for simulating herding behavior in dynamic environments.

Introduction

Herding algorithms, at their core, strive to model the behavior of a shepherd (sheepdog) endeavoring to control larger groups of unwilling individuals (sheep). Traditional approaches rely on predefined rules to govern the individual behavior of agents within a group, influencing their movements based on factors like proximity and alignment with neighbors [1]. While these methods capture the fundamental aspects of collective behavior, their deterministic nature might fall short in handling the inherent uncertainty and imprecision found in real-world scenarios. In this context, our approach introduces another dimension to herding algorithms by incorporating fuzzy logic. This enables a more flexible and adaptive decision-making process by allowing degrees of truth between the absolute true and false, thus enabling the individuals in a group to respond more effectively to altering environments.

As a starting point, we take the algorithm proposed by Strömbom et al. [2], which has already been implemented by our predecessors in a Unity environment. Strömbom's model employs a shepherd alternating between *collecting* dispersed sheep and *driving* the group towards a specific goal. The sheep's movement is determined by a weighted sum of various forces.

In this paper, we discuss our approach to the re-implementation of Strömbom's algorithm and its suggested improvements using fuzzy logic. Additionally, we explore the viability of the sheep's learning behavior through a genetic algorithm, whereby sheep incrementally improve their tactics through evolution.

Methodology

The testing environment consists of an enclosed pasture with a barn at its edge. The N agents (sheep) are randomly placed in the pasture together with a shepherd (sheepdog). The aim of the shepherd is to collect and drive all the sheep into the barn within a limited amount of time. The behavior of each individual agent in the simulation is governed by a selection of algorithms.

Strömbom model. Agents attempt to move away from the shepherd while staying close to their neighbors. Each agent decides on its next action based on its current position, the shepherd's position $\bar{S}$, and the position of its n nearest neighbors $\bar{A}_i$. The agents are attracted to their neighbors' center of mass (LCM) and repelled from other agents within a boundary distance r_a. They are also repelled by the shepherd if it is closer than r_s.

All of these contributions are summed up by the following components:

1. Repulsion from the Shepherd: If the agent is within a distance r_s of the shepherd, it is repelled in the opposite direction $R_i^s = A_i - S$.

2. Attraction towards the local center of mass: The attraction is represented by the vector $C_i = LCM - A_i$, where $LCM = \frac{1}{n}\sum A_i$ for the agent's n nearest neighbors.

A herding algorithm that uses fuzzy logic

Summary of Findings: Recapitulate key findings and their significance.

fuzzy logic | herding algorithm | genetic algorithm

3. Repulsion from Other Agents: If there are k other agents within r_a, the combined repulsion vector is defined as $R_i^a = \sum_{j=1}^{k} \frac{(A_i - A_j)}{|A_i - A_j|}$.

4. Inertia and Noise: In each timestep, the agents maintain their previous direction H_i, and some random noise ϵ is added to account for unpredictability.

Combining all these components, the movement vector in each iteration becomes

$$H_i' = hH_i + cC_i + a\hat{R}_i^a + s\hat{R}_i^s + e\epsilon.$$

On the other hand, the shepherd is governed by a separate set of rules. It dynamically switches between *collecting* and *driving* based on the maximum distance of agents from the center of mass (either global GCM or local LCM). If all agents are within $r_a N^{\frac{2}{3}}$, it attempts to position itself behind the flock to push it towards the goal. If any agent is farther than $r_a N^{\frac{2}{3}}$, the shepherd instead selects the most distant agent and herds it back towards the flock. This means the shepherd's trajectory is always straight toward the desired collecting or driving position. Additionally, some noise is added to the shepherd's movement as well.

Improvements to the model. The base algorithm excels when the shepherd possesses global knowledge and the agents are attracted to the center of mass of at least half of their peers. However, its effectiveness diminishes with limited attraction among nearby agents and when employing only local knowledge for the shepherd. Our goal is to enhance performance with local knowledge and reduce the dependence on agent behavior. We propose two key modifications: avoiding already collected groups and implementing a modified heuristic for selecting the target agent in collecting mode.

1. **Avoidance of already collected groups (previous algorithm with angle calculation):** To avoid breaking up already collected groups of agents during the herding process, a modification is introduced. Instead of taking a straight path towards the furthest agent from the center of mass (GCM/LCM), the shepherd follows an arc around the calculated GCM/LCM. This approach ensures a constant distance from the GCM/LCM, preventing unintended group disruptions. However, this adjustment may extend the time it takes for the shepherd to reach the desired collecting position, particularly in scenarios where the shepherd is considerably distant from the GCM/LCM compared to the collecting position.

2. **Avoidance of already collected groups and obstacles (new version - weighted sum):** When collecting agents, the shepherd avoids disrupting already gathered groups by circling around the herd in an arc. The desired heading is a weighted combination of a direct target vector (H^d) and an arc movement vector (H^a). The arc movement is calculated as

$$H^a = \sum_{i=1}^{n} \frac{1}{(|A_i - S|)^2} \cdot H_{i^\perp}$$

, where S is the shepherd's position, $A_1, \ldots, A_n$ are the positions of known agents, and H^i is a vector perpendicular to $\hat{A} - \hat{S}$ based on the target position relative to the center of mass.

The shepherd's desired heading ($\hat{H}_s$) is computed as:

$$\rho_d \hat{H}_d + \rho_\alpha \hat{H}_\alpha + \rho_o \hat{H}_o$$

, with weights ρ_d and ρ_α determined by the distance (d) to the nearest agent:

$$\rho_d = 1, if \quad d \geq r_r, else \quad \frac{d}{(r_r)},$$
$$\rho_\alpha = 0, if \quad d \geq r_r, else \quad \frac{r_r - d}{r_r}.$$

When distant from agents, the shepherd moves straight towards the target, transitioning to an arc movement within r_r distance of an agent. The r_r parameter is set equal to the r_s parameter of agents, assuming the shepherd adjusts its path upon approaching that distance.

For field navigation, repulsion from obstacles (R_s^o) is added, calculated as:

$$R_s^o = \sum_{i=1}^{j} \frac{1}{|S - O_i|}(S - O_i),$$

where $O_1, ..., Oi$ are obstacle boundary points within r_o distance. This vector is weighted by a small constant ρ_o.

Fuzzy logic. Adding fuzzy logic to the existing algorithms can introduce a more nuanced and realistic decision-making process for determining the behavior of each agent. Instead of using constant values in the existing models, fuzzy logic can be used to fine-tune the values based on more complex and dynamic criteria. The first step is to fuzzify the inputs, i.e., determine the degree to which they belong to each of the appropriate fuzzy sets via a membership function. The next step is to use the fuzzy inputs to evaluate the level of implication. Then, the outputs must be combined in some manner. All of the fuzzy sets representing each force are aggregated into a single set. Finally, the aggregate output is deffuzified into a single number.

Genetic algorithm. The second focus of the article is making sheep dynamics less scripted but learned. We plan to achieve this by representing the sheep as autonomous agents that utilize a genetic algorithm (GA) to incrementally evolve their herding strategies.

GAs are optimization algorithms inspired by the concept of natural selection. The process starts by generating an initial population of potential solutions. Each solution is assessed using a cost function, and solutions with higher fitness are more likely to be selected for reproduction. The selected individuals undergo crossover and mutation, eventually replacing the less fit individuals in the population. This cycle repeats for multiple generations, refining the solutions over time.

Regarding collective behavior, we need to modify the traditional GA by adding some changes presented in Thomas Miconi's "A Collective Genetic Algorithm"[3]. Since we can not evaluate a single agent but rather the population as a whole, we randomly select two sheep from the population and separately remove them to evaluate their impact on the whole herd. We then create an offspring through crossover and mutation. The less useful sheep, as determined by their impact on the collective herding fitness, are then replaced by the offspring. The sheep's genotype will be represented as a vector, so we can also use it as our sheep's decision model. Currently, the plan is to use a simple three-layer artificial neural network. The input layer will consist of an input vector containing information about distances to other sheep, shepherd and the output layer will return the angular direction and speed of the sheep's movement.

The initial suggestion for the cost function is a weighted sum between the time it took for all the sheep to enter the barn and the distances separating them in a herd. The optimization strategy is designed to encourage sheep to form a cohesive herd while simultaneously maximizing their evasiveness. Examples of similar implementations using ANN models can also be observed in simulating zebrafish collective behaviour[4] and collective behaviors of robots[5], albeit using supervised training of the neural network agent's models instead of genetic algorithms.

Bibliography

1. Lien JM, Bayazit O, Sowell R, Rodriguez S, Amato N (2004) Shepherding behaviors in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* Vol. 4, pp. 4159–4164 Vol.4.
2. Strömbom D et al. (2014) Solving the shepherding problem: Heuristics for herding autonomous, interacting agents. *Journal of The Royal Society Interface* 11.
3. Miconi T (2001) A collective genetic algorithm in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation.* pp. 876–883.
4. Beckerleg M, Zhang C (2016) Evolving individual and collective behaviours for the kilobot robot in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC).* pp. 263–268.
5. Cazenille L, Bredeche N, Halloy J (2018) Modelling zebrafish collective behaviours with multilayer perceptrons optimised by evolutionary algorithms. *arXiv preprint arXiv:1811.11040.*