Metaheuristics Using Agent-Based Models for Swarms and Contagion

Douglas de Jesus[1], Lingge Li[2], Daniel Moyer[1], and Junyuan Lin[3]

[1]University of California, Los Angeles
[2]Scripps College
[3]Pepperdine University

August 9, 2013

1 Introduction

Many animals, including humans, congregate in large numbers. These swarms move in complex fashions and seem to behave as a cohesive unit controlled by a single brain, despite being made up of separate individuals. Given the large number of animals in some cases (starlings form flocks of up to 10 million birds), modeling these swarms poses a challenge. Previous researchers have extensively studied swarming behavior but relatively few included emotion in their models. In addition to looking at how swarms move, we seek to understand how information or emotion spreads among the individuals. The spread of information or emotion in turn may account for the movement patterns found in nature. In this paper, we examine the effects of fear on crowd movement.

We used agent-based models to study swarms. An agent-based model sets up a large system of agents which follow certain rules individually. In our case, we have a system of ordinary differential equations where each differential equation describes an agent's behavior. Using agent-based models is a well-established way to study swarming behavior. The models we used are the three-zone model and the particle swarm optimization (PSO). We derived our three-zone model based on previous work by Chuang et al. [3] and Cucker and Smale [4]. Then we implemented the three zone model in C++ using a linked-cell algorithm to increase efficiency. Normally, computing all the pair-wise interactions would take $O(n^2)$ operations. The linked-cell method cuts down the number of comparisons needed by assuming that the effect of interactions are negligible outside of a certain distance. The linked-cell method partitions the space into cells whose width is determined by the cut-off distance, and only compare each particle to the particles in neighboring cells. Our experiments showed that the computational efficiency is better than $O(\log n)$ with the linked algorithm.

Then we applied these models to two real-life problems, one biological and one social. Previous research by Olson et al. [7] used a genetic algorithm to show how flocking behavior could be naturally selected for. We modified the genetic algorithm and studied how swarming behavior and emotional contagion respond under the pressure of predator. Our results suggest that emotional contagion could evolve in gregarious animals.

We also applied Particle Swarm Optimization (PSO) to model emergency evacuation of an airplane, inspired by the recent Asian plane crash. In the PSO algorithm, each particle's position is compared to some fitness function. Each particle moves according to its knowledge of its own previous best position and the group's current best position. We hope that this will lead to increased understanding of how panicked crowds behave in evacuation situations and that this will lead to better, safer designs.

2 The Swarming model with emotion

The three-zone model has been shown to accurately describe the behavior of flocks of birds and schools of fish [2]. The model defines three interaction zones. The outer zone is the zone of attraction, in which animals are drawn to each other. Each animal needs its own space however, so within the inner radius, the zone of repulsion, animals will move away from one another. In the intermediate zone of alignment, animals will match their velocities, so the swarm moves cohesively in one direction.

We adopted the 3-zone model to describe the motion of agents and incorporated emotion. Specifically, we used the D'Orsogna-Bertozzi model with self-propulsion for attraction and repulsion [3] and the Cucker-Smale model for alignment[4]. Under the assumption that agents would move faster when scared, we included emotion in the drag term. We also assumed that agents would adjust conform their emotions with their neighbors' and used an alignment model similar to the Cucker-Smale for emotion.

$$\dot{x_i} = v_i \tag{1}$$

$$\dot{v_i} = \underbrace{\left(4\alpha \cdot \left(\frac{1}{2} + \frac{q}{2}\right) - \beta \cdot |v_i|\right) v_i}_{\text{Asymptotic Velocity}} + \underbrace{\frac{1}{N_\epsilon} \sum_{j=1}^{N_\epsilon} \left(\frac{C_A}{L_A} \cdot e^{\frac{-|\vec{r}_{i,j}|}{L_A}} - \frac{C_R}{L_R} \cdot e^{\frac{-|\vec{r}_{i,j}|}{L_R}}\right) \vec{r}_{i,j}}_{\text{Attraction - Repulsion}} - \underbrace{\left(\frac{1}{1+|\vec{r}_{i,j}|^2}\right) \vec{v}_j}_{\text{Align}} \tag{2}$$

$$\dot{q_i} = \underbrace{\frac{-q_i}{100}}_{\textbf{Decay}} + \underbrace{\frac{1}{N_\epsilon} \sum_{i=1}^{N_\epsilon} \left(\frac{1}{1+|\vec{r}_{i,j}|^2}\right)^{\gamma} (q_i - q_j)}_{\text{Agent to Agent}} \tag{3}$$

3 The Linked Cell Algorithm

While most microscopic particle models include interactions between every pair of particles at every step, it can be computationally expensive to calculate pairwise forces for a large number of points. This problem is O(n^2), and so becomes quite time consuming as n gets large. However, for large distances, the force between points is negligible for many interactions; at long distances, any effect that decays over distance will approach zero as the distance approaches infinity by definition. Thus for a given particle it is reasonable to only calculate the effects from particles within a certain radius [1], in other words only interactions from the ϵ-nearest neighbors.

In order to leverage this, however, we must avoid the ϵ-nearest neighbor search cost, which itself is O(n^2). Towards this end, we implemented the linked cell algorithm described by Quentrec and Brot [8] for agents in two dimensions in a bounded domain.

3.1 The Algorithm

The Linked Cell algorithm is as follows:

1. Parse the domain into a grid of square cells, each with side length equal to the cut off distance (ϵ). Arguments for this choice of side length's optimality are given below.

2. For each cell, instantiate a list of references to agents held within that cell (these will initially be empty).

3. For each agent, add a reference to that agent to the appropriate cell's list; that is, append a pointer with the address of the agent to the list attached to the cell to which the agent should belong.

4. Update each agent based on interactions with any agent in its cell, and all of the cell's immediate neighbors ("left", "right", "up", "down", and all four "diagonals").

3.2 Optimal Cell Length

Consider an agent somewhere within a cell, with side length equal to or greater than the cutoff distance. In order to find all of the agent's ϵ-nearest neighbors, we must search through all of the adjacent cells, but none of the other cells. Thus, up until the side length is equal to the cutoff distance, we may freely decrease the length of side of the cell, thus reducing the search area, without changing our algorithm.

It still should be noted that adding more cells has a memory cost, and thus adding a large number of them may not be feasible for every case and domain. Our implementation uses light cells which carry only references, and thus are easy to traverse. Furthermore, to offset the cost of moving agents from cell to cell, in shared memory we utilize a linked list structure within each cell, to allow easy insertion and removal from each cell. The cost of retrieval from a list is in $O(n)$, opposed to an array, which has $O(1)$, yet as the entire list must be traversed to update the cell this cost is negated.

To see that the cells should be smaller than ϵ, we consider the spatial resolution required in order to search less area. If we take side length equal to $\frac{\epsilon}{4}$, we must search 77 cells (we'll avoid searching the four "corner" cells). While the area of these cells are $\frac{1}{16}$th the size of the cells with side length ϵ, the computational overhead of storing references and looping over each cell will start to eclipse the utility of searching less area. Furthermore, smaller cells means agents will shift cells much more often, which also increases the computational cost.

3.3 Performance

As seen in figure 1, the linked cell algorithm for large numbers of agents scales below $O(n \log(n))$, which was reported in Yao et al [9].

4 Evolution of Emotional Contagion

4.1 Genetic Algorithm

Previous research has used genetic algorithms to show that predator confusion is a sufficient selective force for flocking behavior to emerge naturally. [7] Olson's study had the agent's behavior controlled by hidden Markov networks described by varying length bit-strings. Our work uses the same 3-zone model and predator-prey modifications described earlier in this text. The genes on which we evolved are different coefficients from that model, which take values in [0,1]. Each value is stored as an 8-bit integer, so a particular individual's genome is a 28-bit string.

Genetic algorithms are inspired by nature and are used in optimization problems [6]. They involve three steps: selection, mutation, and recombination. For selection, each potential solution is evaluated using some fitness function. In Olson's study, each genome was used to populate a number of identical prey, which were put into a simulation with predators. The fitness rating would then be how few members died. However, we decided to simplify the fitness test by modelling it after what really happens in predator-prey scenarios in nature. All the current genomes are represented at the same time. The simulation is run until a certain proportion of the starting population has been killed. The fitness test, then, is simply whether or not the animal survived long enough to pass on its genes. We believe this most closely resembles what actually happens in natural selection. The algorithm we used is as follows:

1. Initialize: Create a world with 10 predators and 500 prey with some initial parameters (either random or seeded to some values that were previously determined to work well).

2. Selection: Run the simulation until half the population has been killed.

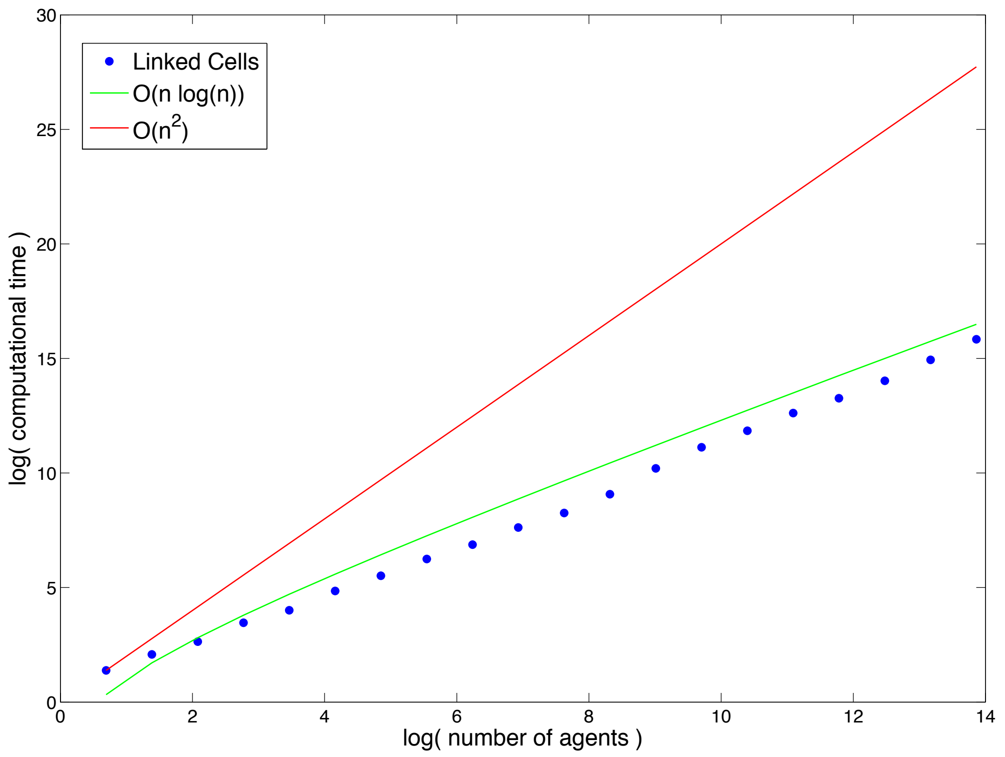

Figure 1: This is a log-log plot of the run time of Linked Cell algorithm as the number of agents increases. Also plotted are two theoretic benchmarks, n^2 and $n\log(n)$.

3. Recombination: Select two random parents. Choose a crossover point x in the range [1,8], then create a child with the first x genes from the first parent and the rest from the second.
4. Mutation: Each gene is mutated with chance 1 in 32. One random bit in the gene is flipped.
5. Do steps 2 to 4 for a fixed number of generations or until convergence is reached.

It is important to note that, in this algorithm, the parents continue on to the next generation.

4.2 The Predator-Prey Model

We added the predator to the previous swarming model with emotion. The predator itself always moves in the direction of the closest prey. Not only do the prey move away from the predator, the predator also invokes fear among the prey. If prey is within a close distance from the predator, it will have a certain probability of being eaten.

Model for the prey

$$\begin{aligned}
\dot{x_i} =& v_i \\
\dot{v_i} =& \underbrace{\left(4\alpha \cdot \left(\frac{1}{2} + \frac{q}{2}\right) - \beta \cdot |v_i|\right) v_i}_{\text{Asymptotic Velocity}} \\
&+ \underbrace{\frac{1}{N_\epsilon} \sum_{j=1}^{N_\epsilon} \left(\frac{C_A}{L_A} \cdot e^{\frac{-|\vec{r}_{i,j}|}{L_A}} - \frac{C_R}{L_R} \cdot e^{\frac{-|\vec{r}_{i,j}|}{L_R}}\right) \vec{r}_{i,j}}_{\text{Attraction - Repulsion}} - \underbrace{\left(\frac{1}{1+|\vec{r}_{i,j}|^2}\right)(\vec{v_j} - \vec{v_i})}_{\text{Align}} \\
&- \underbrace{\frac{C_R}{L_R} \cdot e^{-\vec{r}_p/L_R}}_{\text{Predator}} \\
\dot{q_i} =& \underbrace{\frac{-q_i}{100}}_{\textbf{Decay}} + \underbrace{\frac{1}{N_\epsilon} \sum_{i=1}^{N_\epsilon} \left(\frac{1}{1+|\vec{r}_{i,j}|^2}\right)^{\gamma} (q_i - q_j)}_{\text{Agent to Agent}} \\
&+ \underbrace{\left(\frac{1}{1+|\vec{r}_{i,p}|^2}\right)^{\gamma} \cdot (1 - q_i)}_{\text{Predator}}
\end{aligned}$$

Model for the predator

$$\begin{aligned}
\dot{y_i} =& v_{yi} \\
\dot{v_{yi}} =& -v_{yi} + A\frac{x_j - y_i}{|x_j - y_i|} s.t. |x_j - y_i| = min_{1 \leq k \leq n} |x_k - y_i|
\end{aligned}$$

4.3 The Variable Parameters

We evolved on seven different parameters, labelled a through g in . All parameters are in the range [0,1]. The values were stored as 8-bit integers which were then normalized to fit the range.

$$\begin{aligned}
\dot{x_i} =& v_i \\
\dot{v_i} =& F_{Drag}(f) + b \cdot (c \cdot F_{Attract-Repel} + (1-c) \cdot F_{Align}(g)) \\
&+ (1-b) F_{Pred}(g) \\
\dot{q_i} =& -d \cdot q_i + e \cdot (a \cdot F_{Align}(g) + (1-a) \cdot F_{Pred}(g))
\end{aligned}$$

1. **Attraction/Alignment Ratio**(a): This parameter determines how much an individual's swarming behavior is influenced by the forces of attraction and repulsion versus the force of alignment. A value of 1 means that the individual only follows the rules of attraction and repulsion, and a value of 0 means the individual only aligns its velocity with its neighbors.

2. **Motion Predator/Prey Ratio** (b): This parameter determines how much an individual's acceleration is influenced by its neighbors versus how much it is affected by predators. A value of 1 means that the individual is only affected by the swarming rules, and a value of 0 means the individual only tries to avoid the predator.

3. **Emotional Predator/Prey Ratio** (c): This parameter determines how much an individual's emotional state is influenced by its neighbors versus how much it is affected by predators. A value of 1

means that the individual is only made fearful by its neighbors, and a value of 0 means the individual is afraid of the predators and is not affected at all by its neighbors.

4. **Fear Decay** (d): This parameter determines how quickly an animal returns to the calm state. A value of 0 means that the individual does not calm down on it's own, and 1 means that the individual calms down completely in a single time step.

5. **Empathy** (e): If one individual with fear 1.0 interacts with another with fear 0, it might not be the case that they both tend to a fear value of 0.5. Rather, it could be that the fearful individual has a greater effect on the other's mood, or vice versa. The empathy parameter controls this ratio of how easily one is excited or calmed. A value of 0 means that the individual will only be calmed by its neighbors, and 1 means that the individual will only be made more afraid by its neighbors; a value of $\frac{1}{2}$ would be simple diffusion with conservation of fear.

6. **Adrenaline** (f): In our model, fear influences kinetic behavior by increasing a particle's maximum velocity. This corresponds to the adrenaline rush that an animal gets when stressed. Each particle has a fixed maximum speed that is attained when they are completely afraid, but their default speeds are allowed to vary. The adrenaline parameter represents the ratio of an animal's default, calm speed to their maximum, afraid speed. A value of 0 means that an individual does not move unless afraid, and 1 means that the individual moves its maximum speed at all times.

7. **Gamma** (g) : This determines how quickly the interaction strength drops off as the distance increases in the Cucker-Smale potential. This potential was used not only for velocity alignment but also for fear diffusion as well.Cucker and Smale proved that for $g \leq 1/2$, the flock is guaranteed to converge to a common velocity [4]. A value of 0 is an infinite interaction distance, and 1 is the shortest interaction distance.

5 Results of Evolution

To compare the performance of each algorithm, we ran each using the same parameters for 50 generations. The stochastic method converged in three parameters: gamma and motion and emotion predator-prey ratios. However, the other three parameters still had a high variance. The genetic approach also converged in these parameters to the same values, however it appeared to converge in the other parameters as well. We concluded that the genetic method converges more quickly due to the crossover of genes. Therefore, this was the preferred method for the rest of the simulations.

We performed the following experiments:

1. With all random initial values
2. With predetermined starting values: a = 0.5, b = 0.5, c = 0.6, d = 0.03, e = 0.7, f = 0.5, g = 0.45
3. With fixed f = 0.5
4. With fixed a = 0.5, d = 0.03, e = 0.6, f = 0.5
5. With the following restriction: c + d - f = 1

The following section discusses the optimal values we found and their significance.

5.1 Interaction distance

The interaction distance drop-off g converges to about 0.25, which corresponds to intermediate-to-long range interaction; the force felt at a distance of 75 is half of that felt at 25. The fact that long-range effects are not negligible means that the agent gains some benefit from long-range sensing. Note that the ϵ cutoff for the

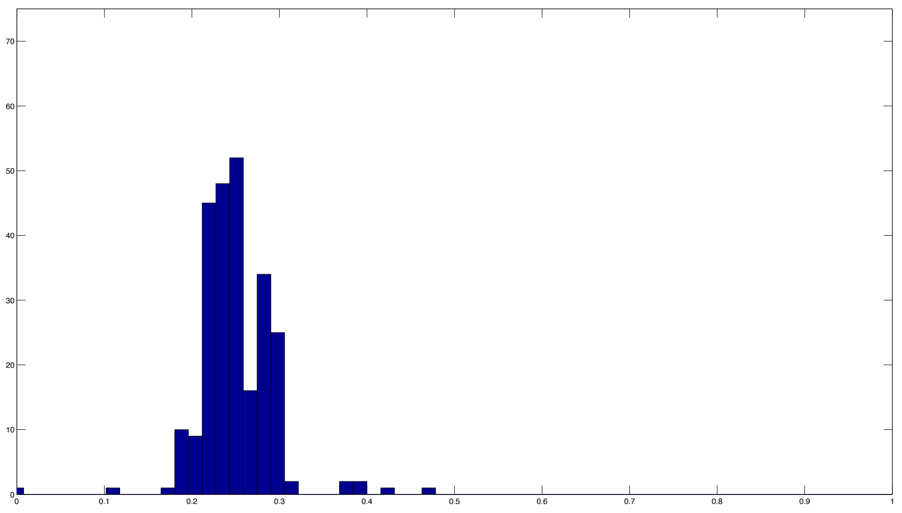

Figure 2: Interaction distance drop-off after 140 generations

linked cells algorithm stays the same, and g only determines the steepness of the alignment function. Thus even if the force is positive at a cell length's distance, no interactions are calculated beyond that distance. At the same time, if the value were any lower, far-away particles would be too influential and the agent's behavior would be hampered by irrelevant information.

5.2 Effects of fear

The fear decay rate d converges to 0.1 The low fear decay rate means that the prey will stay afraid for a while even after they are away from sources of fear. For instance, if a prey was running from a predator and the predator went off in another direction, the prey would continue to be afraid and run quickly. This could help prey continue to avoid predators that might be slightly outside of their detection range.

Empathy e tends to higher values, meaning that fear is more influential than calmness. The high empathy values mimic how we expect emotion to spread in real life. Agents that are afraid are more influential than those who are calm. This also helps counteract the emotional inertia caused by the group. If many prey are in a group and only one of them is within range of a predator, that one will become afraid. However, because emotion spreads from agent to agent, the group acts like a heat sink; it takes a long time for any of them to become afraid. A high empathy means that fear spreads more strongly than calmness, so the latency of the group's fear is negated.

Initial simulations showed adrenaline converging to 1 consistently. This intuitively made sense, as it meant the prey would run their maximum speed all the time. However, after we discovered an error in the code that invalidated the data, we had to disregard these results. After the fix, adrenaline converged to around 0.2 to 0.35 when all parameters were free. But, we had designed an experiment to impose a limited resource on the prey, wherein we assumed that a high adrenaline and low fear decay and emotion predator-prey ratio would be beneficial, since a prey would run its maximum speed indefinitely after encountering a predator. Thus, we imposed the restriction c + d - f = 1. This experiment yielded high values for c, d and f. A prey would be calmed quickly, and would be emotionally affected more by fellow prey than by predators, but these were irrelevant as the high adrenaline means that fear only has a small effect on speed. This high adrenaline value contradicts the first two experiments that suggested adrenaline should be low. Due to the conflicting data, we were unable to conclude what the optimal value for this parameter should be.

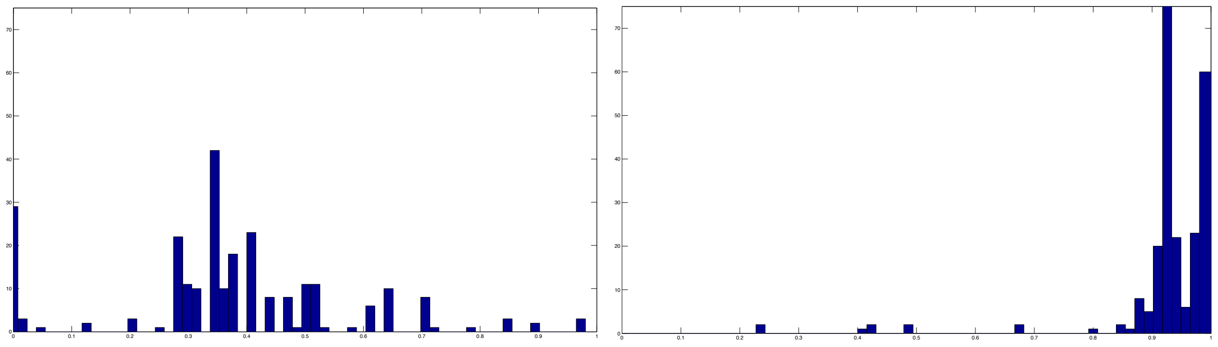

Figure 3: Conflicting results for adrenaline.

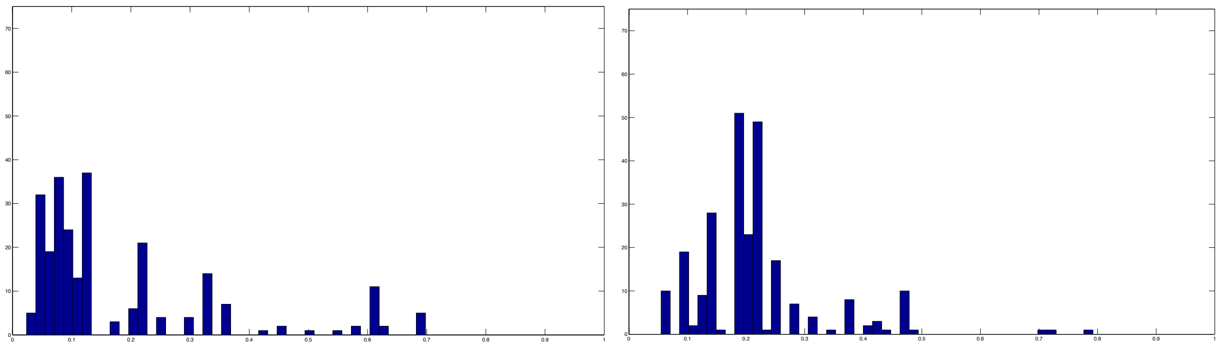

Figure 4: Emotional (left) and kinetic (right) predator vs. neighbor ratios after 130 generations, all other parameters fixed

5.3 Predator vs Neighbor Interactions

The kinetic effect predator/neighbor ratio b converges to about 0.25, while the emotional effect ratio c tends to lower values.These mean that prey tend to value the effects of a predator much higher than the effects of the swarm. Even though there may be many more group members than predators, the predator influence is greater than the combined group effect. This makes sense, as one's survival directly depends on predator avoidance. However, the fact that these values are non-zero means that there is still a survival increase to be gained from following the group.

5.4 Swarming models

The attraction-repulsion model/alignment model ratio a converges to 0.0 - 0.015, which corresponds to only alignment without attraction/repulsion. Our model combines the D'Sorgna-Bertozzi model for attraction and repulsion [3] and the Cucker-Smale model for velocity alignment [4], but we were unsure how strong these should be relative to one another. This is what a refers to. However, the results show that in our model, prey were more likely to survive when they followed only the Cucker-Smale model.

In many simulations, a prey would move towards a nearby group, but in doing so would actually be moving towards a predator that was out of its detection range. It appeared that in this case, swarm attraction was actually detrimental. If, in the same situation, the prey started heading in the same direction as the others, instead of moving towards them, it would likely have survived.

However, it is known that animals do form swarms in nature and follow attraction and repulsion forces.

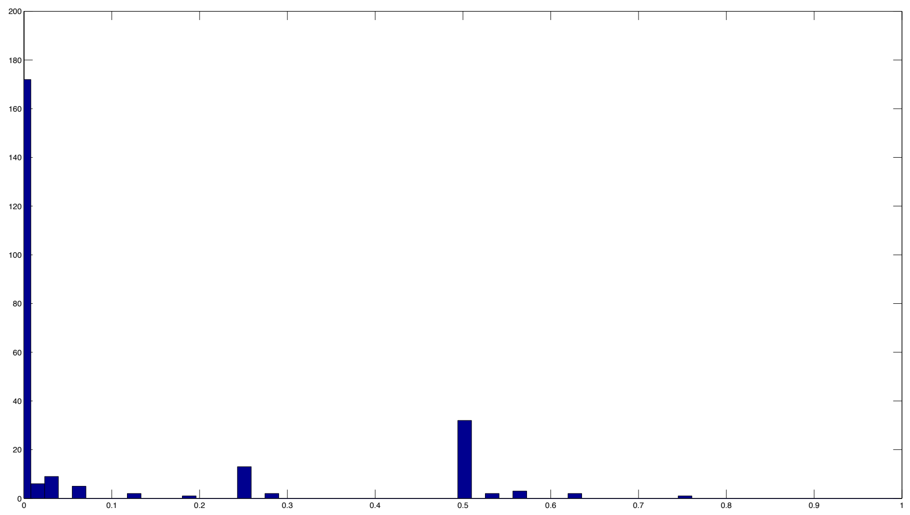

Figure 5: After 145 generations with all other parameters fixed.

In fact, Chuang et al. show that self-propulsion, attraction and repulsion without velocity alignment is sufficient to produce spiral-shaped patterns that are observed in schooling fish [3]. We theorize that the conflict between our results and previous research is a result of the simplifications made in our simulation. In another paper using genetic algorithms on predator-prey behavior, Olson et al. show that predator confusion is a sufficient selective force for swarming behavior to emerge [7]. In that scheme, the kill-success rate is decreased when there are many prey in the predator's visual field. On the other hand, in our model, the predator can eat many prey at once if they are close together. Adding some sort of predator confusion, or a waiting period between kills, would likely produce different results.

5.5 The Stochastic Approach - A Comparison

Inspired by population models, we also experimented a stochastic approach in the genetic algorithm. Instead using genes to control the parameters, we assumed that each parameter is an independent random variable. After each generation, we used histograms to approximate the distributions of the parameters. To match the 8-bit string, we used histograms with 256 bins. Then at the start of the next generation, we used the distributions to generate random numbers from [0,1] as the new set of parameters. We observed that more parameters converge in the genetic method. This may be explained by that some of the parameters are correlated and the genetic method preserves correlations better than the stochastic method. In fact, the stochastic method doesn't preserve any correlations at all because the parameters are assumed to be independent. A possible solution may be using a multivariate distribution but it would be too computationally expensive to approximate with histograms. Thus, we concluded that the genetic approach is better than the stochastic method.

We also observed that the parameters that converge in both methods converge to the same values which further validates our results. One of the parameters is the emotional predator/prey ratio which determines the effects of emotional contagion. The fact that this ratio converges to a non-zero value suggests that natural selection favors emotional contagion. If emotional contagion had no positive effects on survival, the ratio would have converged to zero and the emotion of the prey would be only be affected by the predator.

6 Asiana Flight Simulations

Another scenario which we wished to model was individuals escaping from a bounded area. This directly corresponds with recent world events, specifically the Asiana 214 crash, in which a Boeing 777-200ER crashed while attempting to land at San Francisco International Airport. Out of 291 passengers, approximately 180 sustained injury, but only 3 died. One might imagine this to be a panic scenario, where individuals might be crushed by the pressure of those at the back; thus, we wished to design a model to explore conditions under which escapes may be made (or if not, then which passengers might be in the most danger).

We constructed two separate mathematical models for the event, one in which each passenger knows the optimal path to exit, the other in which agents are only informed of the local space (and gradient), yet may share information between them. The first, based off of last year's Bounded Room solution, computes shortest paths for each agent. The second, inspired by Particle Swarm Optimization, has the particles search the space for an optimal point (the exit).

6.1 Eikonal/Shortest Path Model

In the previous year's work, the researchers made three assumptions about people. They are as follows:

Assumption 1: There is a goal or set of goals that people want to reach.
Assumption 2: There are regions of discomfort defined by function $F(\vec{x})$, such that, if $F(\vec{x}) > F(\vec{x\prime})$, a person would rather move through $x\prime$
Assumption 3: The speed of a given person is dependent upon the density of the people in that person's current region.

These assumptions lead us to the constraints for a shortest path: a person will chooses the path that minimizes the discomfort encountered on the way to the goal position. This can be expressed by

$$S = \min_{f} \int_0^1 C(f(t))||\dot{f}(t)||dt$$

This in turn produces a simple differential model where a particle simply follows its chosen path:

$$\begin{aligned}\dot{\vec{x}}_i &= \rho \cdot \vec{v}_i \\ \dot{v}_i &= \dot{S}_i(\vec{x}) - \vec{v}_i\end{aligned}$$

A multi-particle case uses the same equations, without direct interaction between the particles. Each particle needs its own path, which we'll denote S_i. In the case of a dynamic field, the path must be updated continuously, or in the discrete case, at each step, hence S_i^t replaces S_i.

The field itself may be chosen for each individual situation, but in general will be generated by objects in the simulation or other sources. Currently there are three types of objects: fires, exits, and walls, as well as agents themselves, which are produce fields but are not static. However, the implementation time for a new object (say, perhaps, a first class seat) is relatively low as it should use the same template as the existing three objects. We modeled both the fire and the exit uncomfortability as wide Gaussians, seat uncomfortability as a thin, tall Gaussian (we want it movement through it to be possible, albeit with great discomfort), and agents as small Gaussians. Note that though the exit is a negative Gaussian, we need to avoid having any negative field values (otherwise our shortest path solvers will need to be overly complex). Thus, we add back to everywhere in the domain the magnitude of the exit field at it's minimum.

Our implementation of this model used the numerical engine built for the evolutionary algorithm, with a discretized field over the domain to calculate the field intensity at a sufficiently high resolution. Unlike the evolutionary model, instead of computing agent to agent interactions we use the following method to determine an agent's path:

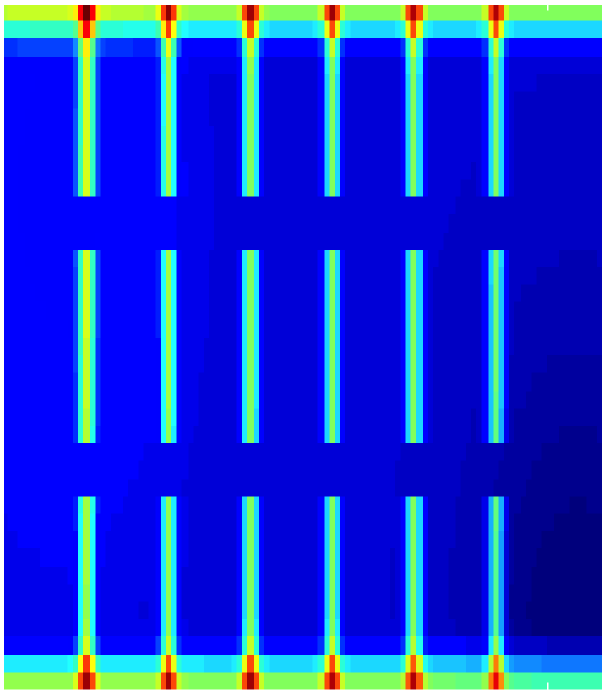

Figure 6: Field values for the specific airplane model. Note that there is a "fire" Gaussian in the upper left corner and an exit in the bottom right, as well as "seats" that are built as low, passable walls.

- Fit a mesh grid of points to the space.
- At each grid point $\vec{x}$, calculate the field strength $F(\vec{x})$.
- For each pair of adjacent grid points, $\vec{x}_i$, $\vec{x}_j$, define edge e_{ij} incident on both points with weight equal to $\frac{F(\vec{x}_i)+F(\vec{x}_j)}{2}$, which is the first order integral approximation.
- For each agent, compute the shortest weighted path from the nearest grid point to that agent to the goal position.

The final two steps are what differentiate this method from the previous year's method. While it may be shown that the solution to the so called Eikonal equation produces the shortest path under our constraints, solving such an equation is not trivial. Though such a solver may produce more numerically accurate results given the same number of nodes, the shortest path method scales much better as nodes are added (Dijkstra's runs in approximately $O(n \log(n))$, and the dynamic field search is $O(n)$ for the number of agents).

This method can be accelerated in practice by using the same grid for each iteration and pre-computing every point's static field value, which is the discomfort field generated by any fixed object. Later, during the computation of the shortest path, any new node encountered may have it's dynamic field value calculated on the fly.

6.1.1 Results and Analysis of the Shortest Path Model

We include, in the accompanying documents and files, a video produced by a simulation using this model. While the transit of multiple passengers down a row is highly improbable in the confined space of an airplane, should the agents "bump" each other to the point at which only one might dominate the lane, we find that they simply stand at an impasse. Both agents have the same force on each other, so neither will be forced out of the aisle. This is not true in reality, and perhaps in future work each passenger could be fitted with a randomly chosen "size".

This model suggests that, should everyone be familiar with the airplane, know where the exits are located and know which exit is the closest to them, the evacuation of the airplane should occur without stampede problems. However, these conditions are rarely satisfied (hence the following section).

6.2 Asiana Flight Simulation using Particle Swarm Optimization

Another approach to simulate the plane crash is inspired by the Particle Swarm Optimization (PSO) algorithm as described in [5]. In nature, herds of animals adapt to their environment by sharing information with each other, and the PSO algorithm provides an effective mathematical model to describe those behaviors. It is widely used to simulate how an initial swarm propagates in the design space towards the optimal solution over a number of iterations (moves). Their moves are based on information about the design space that is assimilated and shared by all members of the swarm[5].

6.2.1 PSO Algorithm

In the PSO algorithm, we are interested in particles' positions and velocities. In our simulation, we first randomly generate initial positions for particles and set their initial velocities to zero. The positions and velocities are denoted x_k^i and v_k^i respectively, where k is the number of iterations and i is the index of particle.

$$x_0^i = \text{Random}$$
$$v_0^i = 0$$

In the next step, we update the velocities of all particles at time $k+1$ and correspondingly their positions using equations from [5].

$$x_{k+1}^i = x_k^i + v_{k+1}^i \, \Delta t$$
$$v_{k+1}^i = \underbrace{w\, v_k^i}_{\text{current movement}} + \underbrace{c_1\, U_k^1\, (p^i - x_k^i)}_{\text{particle memory influence}} + \underbrace{c_2\, U_k^2\, (p_k^g - x_k^i)}_{\text{swarm influence}}.$$

Here,

- w = inertia factor,
- c_1 = self confidence,
- c_2 = swarm confidence,
- p_i = best position of particle i in current and all previous moves,
- p_k^g = position of particle with best global fitness at current move k,
- U_k^1 and U_k^2 are uniform random variables on $[0, 1]$.

Initially, we used the values 0.85, 2.0, 1.0 for w, c_1, and c_2 respectively, and observed that it takes less than 30 iterations for the particles to converge to the optimal point (in our first simulation, we minimized the function $|x - 15| + |y - 20|$).

We also notice that with larger w, particles are more likely to change their velocities even if the better fitness values are discovered. Thus, greater w values are more suitable for searching global fitness. On the contrary, smaller w values correspond to a relatively small possibility for particles to move once a better position is detected, and this behavior is favored when local search is conducted. When increasing c_1, particles trust their own knowledge and move towards their personal best positions so far as opposed to paying attention to the swarm's movement. A larger c_2 corresponds to a bigger opportunity for the swarm to find the local/global optimal position.

6.2.2 Revised PSO Model Without Emotion

We modified the PSO algorithm to better fit real-life scenarios like the Asiana plane crash. Our revised PSO algorithm is:

$$
\begin{aligned}
x_{k+1}^i &= x_k^i + v_{k+1}^i \,\Delta t \\
v_{k+1}^i &= \underbrace{w\, v_k^i}_{\text{current movement}} \underbrace{- \sum_{j \neq i}^{N} \text{Repel}(|x_j - x_i|)}_{\text{collisions}} + \underbrace{c_1\, U_k^1\, (h(\nabla f(x_k^i)))}_{\text{particle local search}} + \underbrace{c_2\, U_k^2\, h(p_k^g - x_k^i)}_{\text{swarm influence}}
\end{aligned}
$$

Here:

- w = inertia factor,
- c_1 = self confidence,
- c_2 = swarm confidence,
- p_k^g = position of particle with best global fitness at current move k,
- U_k^1 and U_k^2 are uniform random variables on $[0, 1]$,
- $h = \frac{x}{const+|x|}$,
- f is the potential function for the plane.

The main differences between this revised version and the original PSO algorithm are reflecting in the repulsion term and the individual particle local search term.

In the PSO model, particles converge very quickly and tend to cluster together, but in reality, people need space and have the tendency to repel other people within certain radius. Therefore, instead of only considering the current motion, particle memory influence, and swarm influence when updating the velocities, we also integrated a particle repulsion term of strength $\frac{1}{r^2}$ where r is the distance between particles.

As for the particle memory influence, we take a different route to calculate the individual particle's local search. We generate a matrix acting as a discrete potential function that describes the geometry of the airplane. (See Figure 6.2.2), having the dangerous region as the global maximum, the chairs being relatively high positions, and exits as global minimums. The seating areas are tilted with the ends that attached to the aisle being lower, and the aisle is a slope that leans towards the exit. The particles' goal is to find the minimum points in designed region. In this way, it is intuitive for the particles to choose their paths. The local search terms for the particles are calculated by finding the gradients of the potential function at their positions. The gradient function is pictured in Figure 6.2.2.

6.2.3 Results

We display still frames from our simulation in Figure 6.2.3. In this simulation, particles' successfully avoid the dangerous area as well as the chairs, and the repulsion force effectively prevents the particles from clashing into each other. One interesting observation is that the last particle to exit the plane is the particle located on the window seat of the row that is nearest to the exit instead of the farthest. A possible explanation of this would be the repulsion force in this simulation is fairly strong, so that the last particle keep being pushed back to the seat. For all of them to escape from the plane, it takes about 1400 iterations.

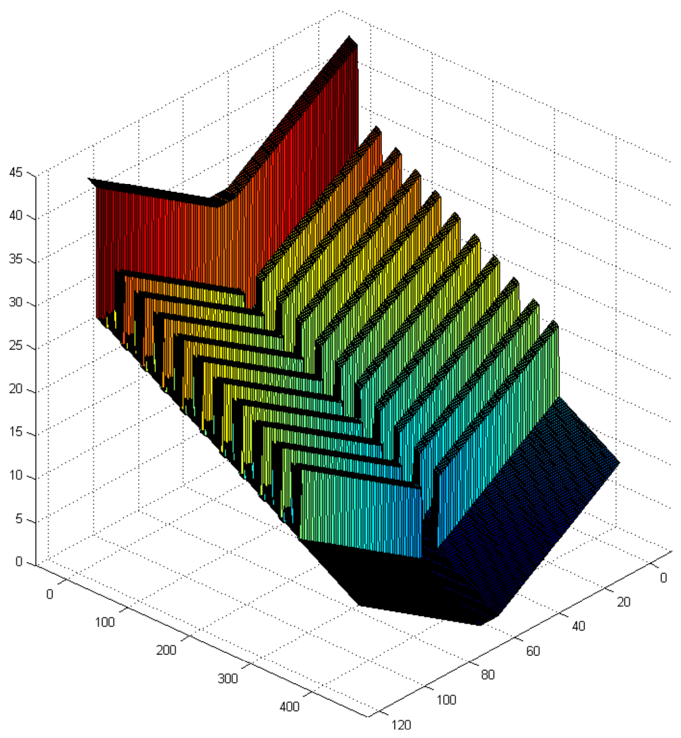

Figure 7: Potential function for the airplane

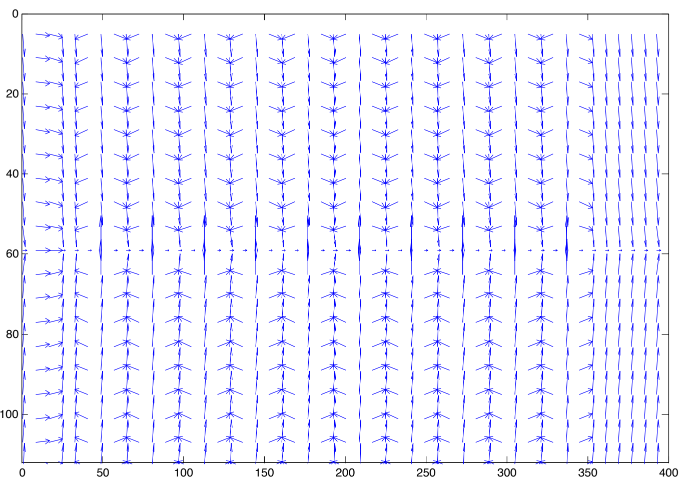

Figure 8: Gradient field corresponding to the potential function

6.3 Revised PSO Model With Emotion

Besides having the particles affect each other's motions, we also consider how emotion spreads among the members of the swarm. Thus, we add a third quantity, q^i, representing emotion value of particle i. Emotion is designed to be a value between zero and one; one being extremely afraid and zero being very calm. The update rule for q^i is identical to the emotion updating rules in the predator-prey model we described before.

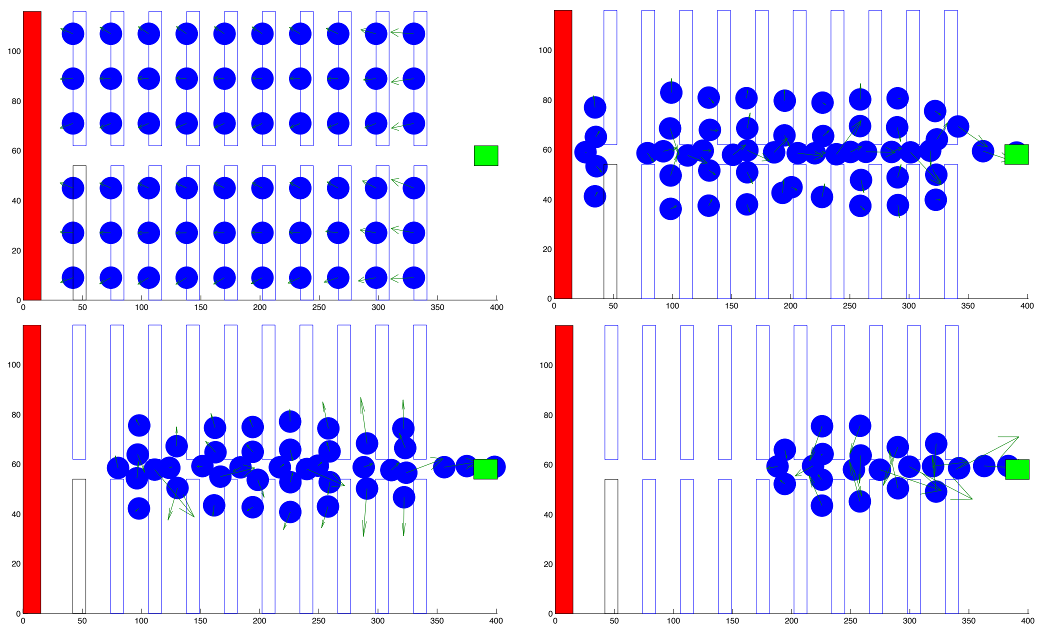
Figure 9: Snapshots of the plane simulation with no emotion at $t = 1$, 200, 400 and 800 iterations.

When an individual has really high q value, they tend to push others harder and urge them to go to the global best position faster than others.Particles with high emotion are also less obstructed by physical barriers such as the seats. Our revised algorithm with emotion is.

$$x^i_{k+1} = x^i_k + v^i_{k+1}\,\Delta t$$

$$v^i_{k+1} = \underbrace{w\,v^i_k}_{\text{current movement}} \underbrace{-\sum_{j\neq i}^{N} \text{Repel}\frac{|x_j - x_i|}{1+(q_j - q_i)}}_{\text{collisions}} + \underbrace{c_1\,U^1_k\,h(\nabla f(x^i_k))}_{\text{particle local search}} + \underbrace{c_2\,U^2_k\,h(p^g_k - x^i_k)\,(1+q^i_k)}_{\text{swarm influence}}$$

$$q^i_{k+1} = q^i_k + \frac{(\text{accumulated emotion change})}{N}$$

Where:

- Repulsion term also depends on differences between two particles' emotion,
- w = inertia factor,
- $h = \frac{x}{const+|x|}$,
- c_1 = self confidence, c_2 = swarm confidence,
- p^g_k = position of particle with best global fitness at current move k,
- U^1_k and U^2_k are uniform random variables on $[0, 1]$,
- f is the potential function for the plane,
- N is the swarm size.

6.4 Result

Compared to particles' behaviors from the non-emotional PSO model, particles near the spreading fire in the emotional PSO simulation significantly increase the speed and rush directly to the global fitness point. This speed enables them to climb over the sub-peaks (chairs), which is likely to happen in reality. It takes around 960 iterations for the swarm to evacuate the aircraft with emotion factor involved. The still frames from this simulation in following Figure 6.4. The red area in the figure represents the fire that spreads over time.

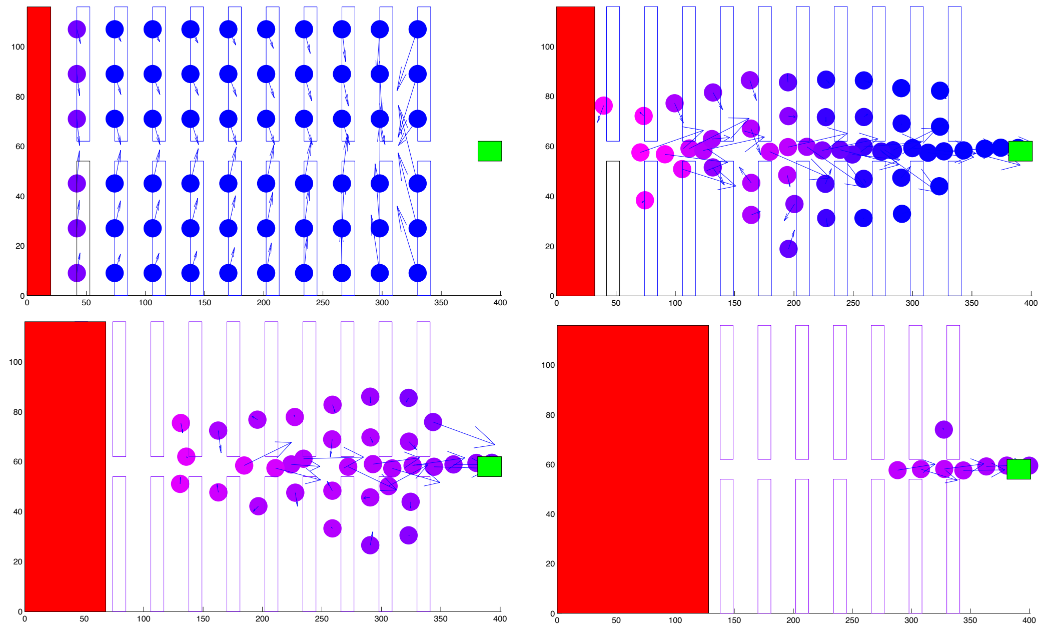

Figure 10: Snapshots of the plane simulation with emotion at $t = 1$, 200, 400 and 600 iterations.

7 Conclusion

We derived a realistic model for swarming behavior with emotional contagion based on two previous well-studied model: the D'Orsogna-Bertozzi model and the Cucker-Smale model. Despite its complexity to study analytically, our model's qualitative behavior can be studied through numerical simulation. Our implementation of the model with the linked cell algorithm significantly increased computational efficiency for large scale simulations. In addition, the simulation platform we built is highly versatile for general agent based modeling.

One application of our model is to mimic the evolutionary process in the predator-prey scenario. In particular, we were interested in the role of emotional contagion and its possible evolution. We added seven parameters and showed that the parameters converge to non-zero values. This suggests that in this model, fear contagion is favored by natural selection.

We then modified the numerical simulation to produce simulations of crowds exiting a room, specifically modelling the Asiana plane crash and informed passengers, as well as implementing a completely different model for uninformed passengers.

We hope to, in the future, continue this project in order to further explore both the evolutionary model produced, as well both of the plane crash models.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids.* Oxford University Press, 1987.

[2] Alethea B.T. Barbaro, Kirk Taylor, Peterson F. Trethewey, Lamia Youseff, and Bjrn Birnir. Discrete and continuous models of the dynamics of pelagic fish: Application to the capelin. *Mathematics and Computers in Simulation*, 79(12):3397 – 3414, 2009.

[3] Yao-li Chuang, Maria R. D'orsogna, Daniel Marthaler C, Andrea L. Bertozzi, and Lincoln S. Chayes. State transitions and the continuum limit for a 2d interacting, self-propelled particle system. *Physica D*, 232:33–47, 2007.

[4] Felipe Cucker and Steve Smale. Emergent behavior in flocks. *IEEE Transactions on Automatic Control*, 52:852–62, May 2007.

[5] R. Hassan, B. Cohanim, O. De Weck, and G. Venter. A Comparison Of Particle Swarm Optimization And The Genetic Algorithm. In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pages 1–13, 2005.

[6] John H. Holland. Genetic algorithms: Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, 267:66–72, July 1992.

[7] Randal S. Olson, Arend Hintze, Fred C. Dyer, David B. Knoester, and Christoph Adami. Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85):., 2013.

[8] B Quentrec and C Brot. New method for searching for neighbors in molecular dynamics computations. *Journal of Computational Physics*, 13(3):430 – 432, 1973.

[9] Z. Yao, J.M. Wang, and Cheng Gui-Rong Liu Min. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Computer Physics Communications*, 161:27–35, 2004.