Modulation of Force Vectors for Effective Shepherding of a Swarm: A Bi-Objective Approach

Hemant Singh*, Benjamin Campbell†, Saber Elsayed*, Anthony Perry†, Robert Hunjet† and Hussein Abbass*
*School of Engineering and Information Technology, University of New South Wales, Canberra ACT, Australia.
Emails: {h.singh, s.elsayed, h.abbass}@adfa.edu.au
†Defence Science and Technology, Australian Department of Defence, Edinburgh SA, Australia.
Emails: {benjamin.campbell, anthony.perry, robert.hunjet}@dst.defence.gov.au

***Abstract*—In the shepherding problem, an external agent (the shepherd) attempts to influence the behavior of a swarm of agents (the sheep) by steering them towards a goal that is known to the shepherd but not the sheep. The problem offers a level of abstraction for Human-Swarm Interaction, where the human is able to shepherd the swarm towards a goal. Similarly, a smart robot could act as a shepherd to replace biological shepherds with ground or air vehicles. In both cases, it is important to preserve the energy/power of the shepherd by modulating the shepherd's influence vector on the sheep. Therefore, in this paper, we design a force modulation function for the shepherd agent to optimize the power used by the agent and systematically study the effect of modulating the force of the influence vector on task success and power used. The problem is further investigated using a bi-objective optimization formulation, where the power used by the shepherd as well as the time of completion of the task are minimized, subject to a threshold of success rate. The findings demonstrate the coupling between contextual information used by the shepherd to modulate its influence vector and the effectiveness and efficiency of shepherd to complete the task.**

***Index Terms*—Force Vector Modulation; Human-Swarm Interaction, Influence; Shepherding; Swarm Robotics**

I. Introduction

The shepherding problem for multi-agent systems has been addressed by a number of authors with a great deal of variation in models for both the shepherds and sheep. For models of sheep, the most common approach is BOIDS [1], [2], [3], [4], [5], [6]. This approach sees the sheep as a BOIDS swarm with the three classic rules of separation, cohesion and alignment.

However the BOIDS model does not describe the behavior of the sheep once they reach a destination, leaving the interpretation/choice of this behavior up to the individual researchers. This results in variations in successful shepherding behavior between approaches. For example, Lien et al. [7] assume that once individual sheep enter the target area, they will not leave. This allows for behaviors where shepherds can steer small groups or individuals to the destination, then leave to collect another group without worrying about dispersion of previously collected groups. In contrast in Lee and Kim [8], there is no such assumption resulting in shepherds being required to stay with sheep that have entered the target area or risk them wandering out.

Modeling of the environment also varies greatly between different studies, where variations include models that take obstacles into account [8], [9] and those that do not [3], [1], target areas that are objects or points [10], circular areas [11] or simply the corner of a "paddock" [3].

The complexity of the problem influences the development of the shepherding behavior. Linder and Nye [12] investigated the implications of the problem design on the utility of the shepherding behavior solution. They concluded that over-simplification of the problem, especially in learning systems, can result in shepherding behaviors that have little a generic value.

As multi-agent shepherding moves from simulation to real-world applications on robotic platforms, the limitations of the robotic platforms will need to be considered in the design of the shepherding logic. An important constraint on any mobile platform is energy storage and the rate at which that energy is consumed. Although there have been some successful demonstrations of herding behavior on real systems [13], [14], [15], the closest literature has come to addressing energy efficiency in shepherding systems is in Tsunoda et al. [16], where they investigate the efficiency of movement of shepherding agents by manipulating repulsion/attraction zone sizes. However, no literature was found that specifically aims to improve the energy efficiency of shepherd behavior.

In this paper, we explore a dimension of shepherding that was not explored in the existing literature: modulation of force vectors. In practice, the sheepdog displays different speeds and sometimes even a lying down behavior. This "resting" behavior, be it in the form of speed control or coming to a complete stop presents as a gap in the current literature; we hypothesize it has a profound impact on task success and energy consumption, especially when the shepherd is a robot vehicle. By affording the shepherd the ability to self-modulate its force vector, it could adapt its behavior and save energy. Furthermore, in order to obtain good values of the modulation parameter, we formulate a bi-objective optimization problem that attempts to minimize energy as well as the time of completion of the task, while ensuring that a certain level of success rate is met.

In addition, in order to investigate the relationship between energy efficiency and shepherding success, a modification of the model described by Strömbom et al [3] has been developed which uses a novel force modulation function for the shepherding agent to allow the shepherding agent to control its velocity in order to provide trade-off between energy consumption and task completion time. The Strömbom model was selected as it uses a simple and well-known model for agent movement, can be applied to single or multi shepherd systems and its use of decentralized control, sensed data and local knowledge matches the realism of physical full-distributed systems. Furthermore, it has been validated for its biological plausibility and ability to mimic the behavior of real shepherds.

The remainder of the paper is organized as follows. We start with a literature review of the shepherding to showcase the variety of approaches used and establish the research gap that the current paper addresses in Section II. We present the simulation model and introduce the function for self-modulation of the shepherd's force vector in Section III. Experimental design, results and discussions are then presented in Sections IV and V, respectively, and conclusions are then drawn in along with future work in Section VI.

II. Background

Although the literature on shepherding in the multi-agent system domain is limited to a dozen papers, the work has been diversified due to the importance of the problem in the simultaneous control of multiple agents with one or a small number of operators/shepherds.

Most shepherding models in the literature employ a single shepherd to control the sheep swarm; especially the learning system literature, this is likely due to the complexity of learning over multiple shepherds simultaneously [17], [12], [13]. Scripted shepherds also relied on a single shepherd, either as a stepping stone to multi shepherd systems [7], [1], or due to the complexity of real world robotic systems [4].

Learning behavior for shepherding has been demonstrated using reinforcement learning [17], [18], genetic algorithms [19], [13], co-operative co-evolution [20] and evolutionary neural networks [12]. In the majority of the learning system literature, the goal is to demonstrate learning systems that can learn the shepherding task in the minimum number of training simulations. An exception to this is Ozdemir et. al. [21], which aims to demonstrate shepherding behavior using the simplest agents possible. Ozdemir et. al successfully demonstrated in simulation that evolutionary methods could be used to generate a control function for shepherding agents without memory, computing or communication resources that resulted in co-operative shepherding behavior that was scalable from 10 to 40 shepherd agents and 10 to 100 sheep.

In contrast, in the pre-programmed shepherding literature, the goal is in defining and demonstrating optimal shepherding behavior. Shepherding can be broken into 3 main subtasks "herding", "collecting" and "patrolling" [8], [7]. In herding tasks the shepherd attempts to move sheep to a specific area or point and in collecting the shepherd attempts to move the individual sheep into a controllable herd.

Herding and collecting are referred to as "driving" and "collecting" by Strömbom et al. [3]. The literature has rich models to describe driving including driving straight at individual sheep [5], or the center of mass of the flock [3], moving side to side [7], or in a V-formation [2], [1]. Collecting appears to be much more limited in terms of available strategies, with behavior dictated by the ratio of shepherds to sheep [8].

III. Methodology

In this section, we firstly describe the Strömbom et al. [3] model for shepherding then present our proposed modulation model for the shepherd influence force vector.

We denote the set of sheep agents with $\Pi = \{\pi_1, \ldots, \pi_i, \ldots, \pi_N\}$, where the letters Π and π are chosen as the first character of the Greek word for sheep $\Pi\rho\acute{o}\beta\alpha\tau o$ and denote the set of shepherd agents with $B = \{\beta_1, \ldots, \beta_j, \ldots, \beta_M\}$, where the letters B and β are chosen as the first character of the Greek word for Shepherds $B o\sigma\kappa\acute{o}\varsigma$. We denote the set of behaviors in the simulation with $\Sigma = \{\sigma_1, \ldots, \sigma_K\}$, where the letters Σ and σ are chosen as the first character of the Greek word for behavior $\sigma\upsilon\mu\pi\epsilon\rho\iota\varphi o\rho\acute{\alpha}$.

Agents are initialized in a square area. We use u to denote the unit, where u is a meter in Strömbom et al. [3]'s original model but could equally generalize to other units per the application. The agents adapt different behaviors described below.

1) Shepherd β_j driving behavior: When the sheep is clustered in one group, the shepherd drives the sheep towards the goal by moving towards a driving point that is situated behind the sheep on the ray between the sheep center of mass and the goal. The shepherd moves towards the driving point with normalized force vector, $F^t_{\beta_j cd}$.
2) Shepherd β_j collecting behavior: If one of the sheep is further away from the group, the shepherd drives to a collection point behind this sheep to move it to the herd; in other words, to collect it. The shepherd moves to the collection point with a normalized force vector $F^t_{\beta_j cd}$. A visualization of driving and collecting behavior discussed above can be found in [3].
3) Shepherd β_j adds a random force, $F^t_{\beta_j\epsilon}$, at each time-step to help resolving deadlocks. The strength of this angular noise is denoted by $W_{e\beta_j}$.
4) Shepherd β_j total force $F^t_{\beta_j}$ is then calculated as:

$$F^t_{\beta_j} = F^t_{\beta_j cd} + W_{e\beta_j} F^t_{\beta_j\epsilon} \tag{1}$$

5) Sheep π_i repulses from β_j using a force $F^t_{\pi_i\beta}$.
6) Sheep π_i repulses from other sheep $\pi_{i1}, i1 \neq i$ using a force $F^t_{\pi_i\pi_{i1}}$.
7) Sheep π_i is attracted to the center of mass of its neighbors $\Lambda^t_{\pi_i}$ using a force $F^t_{\pi_i\Lambda^t_{\pi_i}}$.
8) Sheep π_i angular noise uses a force $F^t_{\pi_i\epsilon}$.
9) Sheep π_i total force is calculated as:

$$F^t_{\pi_i} = W_{\pi_\upsilon} F^{t-1}_{\pi_i} + W_{\pi\Lambda} F^t_{\pi_i\Lambda^t_{\pi_i}} + W_{\pi\beta} F^t_{\pi_i\beta_j} +$$

$$W_{\pi\pi} F^t_{\pi_i\pi_{-i}} + W_{e\pi_i} F^t_{\pi_i\epsilon} \tag{2}$$

where each W representing the weight of the corresponding force vector.

The total force of each agent is used to update the agent position as depicted in Equations 3 and 4.

If there is a sheep within three times the sheep-to-sheep interaction radius, the agent will stop; thus, it will set its speed to zero: $S^t_{\beta_j} = 0$, otherwise it will use its default speed, $S^t_{\beta_j} = S_{\beta_j}$. The speed of a sheep is assumed constant; that is, $S^t_{\pi_i} = S_{\pi_i}$

$$P^{t+1}_{\pi_i} = P^t_{\pi_i} + S^t_{\pi_i} F^t_{\pi_i} \tag{3}$$

$$P^{t+1}_{\beta_j} = P^t_{\beta_j} + S^t_{\beta_j} F^t_{\beta_j} \tag{4}$$

Both π and β agents in Strömbom model move with fixed speed. Generally, this is neither biologically plausible since dogs, for example, do not move with a constant speed all the time they are shepherding, nor it is technologically appropriate when considering the fact that as the approach and closing speed of a vehicle shepherd on sheep represents a key behavioral attribute (for example, aggressiveness) that influence both the effectiveness and efficiency of successful shepherding.

Smart π and β agents in a real-world robotic setting need to comprehend their environment and make appropriate decisions on how to modulate their speed in response to the state and changes of their internal and external contextual information. In this paper, we focus on modulating the speed of the shepherd. The model is generalizable to all types of agents in the model. However, modulating all agents simultaneously will generate complex coupling in the dynamics. Focusing this paper on modulating the shepherd alone is important to systematically understand the effect of speed modulation before it would be possible to consider the coupling effect between concurrent modulation of speed between shepherds and sheep.

To avoid a sudden change in modulation, we require the modulation function to be a smooth continuous function. Being smooth will be helpful for applying optimization methods to optimize the way modulation takes place. Moreover, the function needs to be bounded. The class of logistic functions offers one way to do the modulation that satisfies these conditions. We denote the logistic modulation function for shepherd β_j by $M(S_{\beta_j}, \alpha_{\beta_j}, m^0_{\beta_j}, m^{\sigma_{10}}_{\beta_j})$ as follows:

$$M(S_\beta, \alpha_{\beta_j}, m^0_{\beta_j}, m^t_{\sigma_{10}\beta_j}) = \frac{1}{1 + e^{-\alpha_{\beta_j}(m^t_{\sigma_{10}\beta_j} - m^0_{\beta_j})}} \tag{5}$$

The steepness of the function, α_{β_j}, is an agent's trait used to control how the agent modulates its own speed, and is the focus of the analysis in this paper. The Center of the function is an agent β_j specific value. The input is both agent β_j and time t specific, where σ_{10} denote the index of the modulation behavior in the set of behaviors Σ.

The last input to the modulation function above, $m^t_{\sigma_{10}\beta_j}$, is in principle the output of the fusion function $C(t, \beta_j)$ that integrates contextual information to decide on how modulation occurs.

The speed of shepherd β_j at time t is modulated as follows:

$$S^t_{\beta_j} = S_\beta * M(S_\beta, \alpha_{\beta_j}, m^0_{\beta_j}, m^t_{\sigma_{10}\beta_j}) \tag{6}$$

The acceleration, $\mathfrak{a}^t_{\beta_j}$ meter per second-square, of shepherd β_j at time t is calculated as

$$\mathfrak{a}^t_{\beta_j} = \frac{S^t_{\beta_j} - S^{t-1}_{\beta_j}}{\delta_{\beta_j}}$$

where $\delta_{\beta_j} = \frac{1}{\mathbb{S}_{\beta_j}}$, $\mathbb{S}_{\beta_j}$ is the sampling frequency per second used by the β_j agent to sense and act in the environment; in other words, it is how frequent the agent needs to sense the environment and produce an action. We assume that the sampling frequency of the decision making model is slow enough for any requested change in speed of an agent to take effect. A friction force could be incorporated in Equation 1 if necessary.

Given the sampling frequency above, the shepherd agent will move a distance $\mathfrak{D}^t_{\beta_j} = \delta_{\beta_j} S^t_{\beta_j}$; hence the amount of work, $\mathfrak{W}^t_{\beta_j}$, done by agent β_j at t is

$$\mathfrak{W}^t_{\beta_j} = F^t_{\beta_j} * \mathfrak{D}^t_{\beta_j} \tag{7}$$

and the amount of power $\mathfrak{P}^t_{\beta_j}$ is

$$\mathfrak{P}^t_{\beta_j} = \mathbb{S}_{\beta_j} \mathfrak{W}^t_{\beta_j} \tag{8}$$

IV. Experimental design

The thesis of this paper is anchored on the question on how to design the modulation behavior of a shepherd to its force vector to optimize its energy consumption while completing the task successfully and in the shortest possible time. Mathematically, the optimization problem is to find α_{β_j} that minimizes task completion time and ensure that the task is successful. This latter condition is a hard constraint defined as shown in Equation 9, whereby the task is announced to be completed when the distance between the position, $P^t_{\pi_i}$, of each sheep, π_i, and the position of the goal, P^t_G, is less than a predefined threshold. It is worth noting that in the discrete state space, the circular topology of the Euclidean distance is replaced with a grid.

$$\forall \pi_i, \|P^t_{\pi_i} - P^t_G\| \leq \mathbb{D} \tag{9}$$

Given that with a single shepherd, the above optimization problem is in a single variable, we will explore the problem space through a series of experimental studies to understand the trade-off between energy consumption and task completion time under different modulation strategies. The input to the modulation function is the distance between the shepherd and closest sheep to the shepherd; shown in Equation 10.

$$m^t_{\sigma_{10}\beta_j} = \arg\min_i \|P^t_{\beta_j} - P^t_{\pi_i}\| \tag{10}$$

We use a number of performance measures to analyze the results such as the time taken by a shepherd to successfully complete the task, the power needed by the shepherd to complete the task, and the success rate of the shepherd in completing the task.

All simulations are run for a number of simulation steps per the equation suggested by Strömbom et al. [3] model. We experimented with 80, 120 and 180 sheep and three different initialization methods for sheep location: as a single cluster north to goal (Init 1), two clusters one at North East and North West (Init 2), and three clusters at North, North East, and North West (Init 3). A factorial design generated nine different initializations, each of which was run for 10 simulations to estimate values of the three performance measures. The shepherd was always initialized 50m South of the goal. Each initial sheep cluster was initialized 75 meters away from the goal at the direction of the cluster, with all sheep initialized within a radius of 30 meters. The environment is assumed to be infinite with no boundary. However, for visualization purposes in this paper, all figures of the environment use $150m \times 150m$ area centered on the location of the goal. The task is assumed to be completed when all sheep is within distance $20m \times 20m$ of the goal.

The parameters of the Strömbom model were set to their default values as presented in the original paper. Strömbom experimentally induced that all runs will be successful if $0.53N \leq \Omega_{\pi_i \pi} \leq N$, where $\Omega_{\pi_i \pi}$ represents the number of π agents (neighborhood) a π_i agent operates on. Accordingly, we used $\Omega_{\pi_i \pi} = 0.8N$.

V. Results

A. Sensitivity Analysis

The first set of results aimed at comparing the baseline model without modulation against a model with modulation using a steepness $\alpha_{\beta_j} = 2$ to understand the effect, if any, of modulation. As shown in Table I, modulation is faster and saves power for herd sizes of 80 and 120, regardless of the initial distribution of the sheep. In these cases, the sheepdog managed to steer the sheep successfully to the goal in all runs.

When the size of the herd increases to 180, success rate drops. Modulation had a better success rate with the first initialization and worse with the other two. In the case of the first initialization, modulation also used less power and completed the task faster. We hypothesize that modulation is undesirable when task success is low. The sheepdog needed to use full power to increase task success. However, when task success was high, modulation was able to manage the power to its optimum level; thus, identifying wasted efforts and reduced them. In essence, optimizing efficiency (time, power) is more meaningful when effectiveness (success) is high and less meaningful when effectiveness is low.

The first limitation in the above piece of results stem from the fact that we fixed the value of steepness $\alpha_{\beta_j} = 2$. To understand the effect of the steepness of the modulation function, we varied α_{β_j} between 0 (which results in half the maximum speed independent of context) to 10 in a step of 1.

TABLE I
A comparison between the model without modulation as a baseline and the model with modulation with a steepness $\alpha_{\beta_j} = 2$

No Sheep	Without Modulation			With Modulation		
	Init 1	Init 2	Init 3	Init 1	Init 2	Init 3
Percentage of Average Completion Time Relative to Baseline						
80	100%	89%	92%	**97%**	**87%**	**91%**
120	100%	85%	88%	**95%**	**84%**	**87%**
180	100%	**103%**	**99%**	**94%**	128%	118%
Percentage of Average Power Relative to Baseline						
80	100%	88%	91%	**97%**	**87%**	**90%**
120	100%	85%	88%	**95%**	**84%**	**87%**
180	100%	**103%**	**99%**	**94%**	128%	118%
Success Rate						
80	100%	100%	100%	100%	100%	100%
120	100%	100%	100%	100%	100%	100%
180	60%	**60%**	**60%**	**70%**	40%	50%

The success rate was 100% for a swarm size of 80 and 120 independent of the value of the steepness factor. This was not the case for a swarm size of 180. We show the case of swarm size of 180 in Table II. As shown, the steepness factor has a multi-modal effect on success rate. Different initial dispersion of a swarm requires different settings for the steepness factor. For initialization methods 2 and 3, it is possible to almost double the success rate of each scenario by setting an appropriate steepness factor.

TABLE II
Success rate for a swarm size of 180 with different steepness factor and initialization methods.

α_{β_j}	Init 1	Init 2	Init 3
0	60%	70%	90%
1	50%	40%	40%
2	70%	40%	50%
3	50%	60%	60%
4	30%	20%	40%
5	60%	70%	80%
6	60%	50%	50%
7	20%	20%	50%
8	50%	30%	40%
9	40%	50%	50%
10	50%	30%	50%

The minimum power occurred in all scenarios when $\alpha_{\beta_j} = 0$, which is logical as this values halves the speed of the shepherd over the course of a simulation. As shown in Table II, when the speed is halved for the third initialization, the success rate increased substantially. There also seemed to be a relationship between the task completion time and average power consumption.

To better see the trade-off between success rate and power, we visualize the two values for different value of α_{β_j} in Figure 1. It is important to mention that we excluded here the case of $\alpha_{\beta_j} = 0$ because it is equivalent to a no-modulation case with lower speed. The lowest power was achieved at values 2, 5, and 5 for the first, second, and third initializations, respectively. The fact that the lowest average power for $\alpha_{\beta_j} > 0$ also occurred for the largest success rate seemed initially counter-intuitive because as the agent modulates its

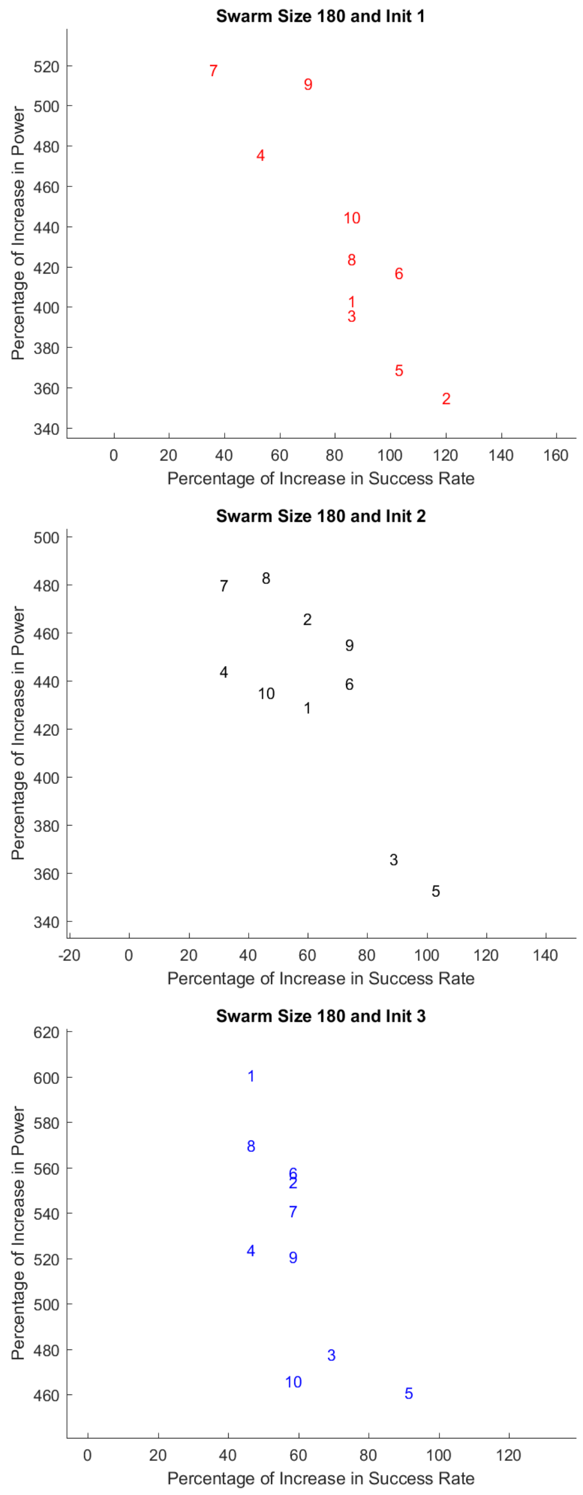


Fig. 1. The trade-off between success rate and power consumption for a swarm size of 180 and the three different initializations. The value of α_{β_j} is shown in each graph as a percentage of $\alpha_{\beta_j} = 0$. Best results are with high success rate (x-axis) and low power consumption (y-axis).

speed to a slower speed, it may take longer to complete the task; thus some runs may end up being unsuccessful. However, after reflection, as the modulation of the force vector becomes more effective in increasing the success rate, it is this form of modulation that also saves power and, as such, the average power saved is higher for a higher success rate. Moreover, the way a sheepdog approaches the swarm impacts the movement of the swarm; which implies that, sometimes approaching the swarm with less force would have better positive effect than maintaining a constant force. Nevertheless, such an explanation is not always consistent as shown in Figure 1, where different values of α_{β_j} resulting in the same success rate had different level of power associated with them. For example, for the first initialization, both $\alpha_{\beta_j} = 5$ and $\alpha_{\beta_j} = 6$ had the same success rate, but different powers.

B. Bi-objective Optimization

The above results suggest that the value of α_{β_j} needs to be optimized for each setup independently. The discrete cases we explored above with the sensitivity analysis and the constraint in the sensitivity analysis that α_{β_j} was always an integer value are then relaxed.

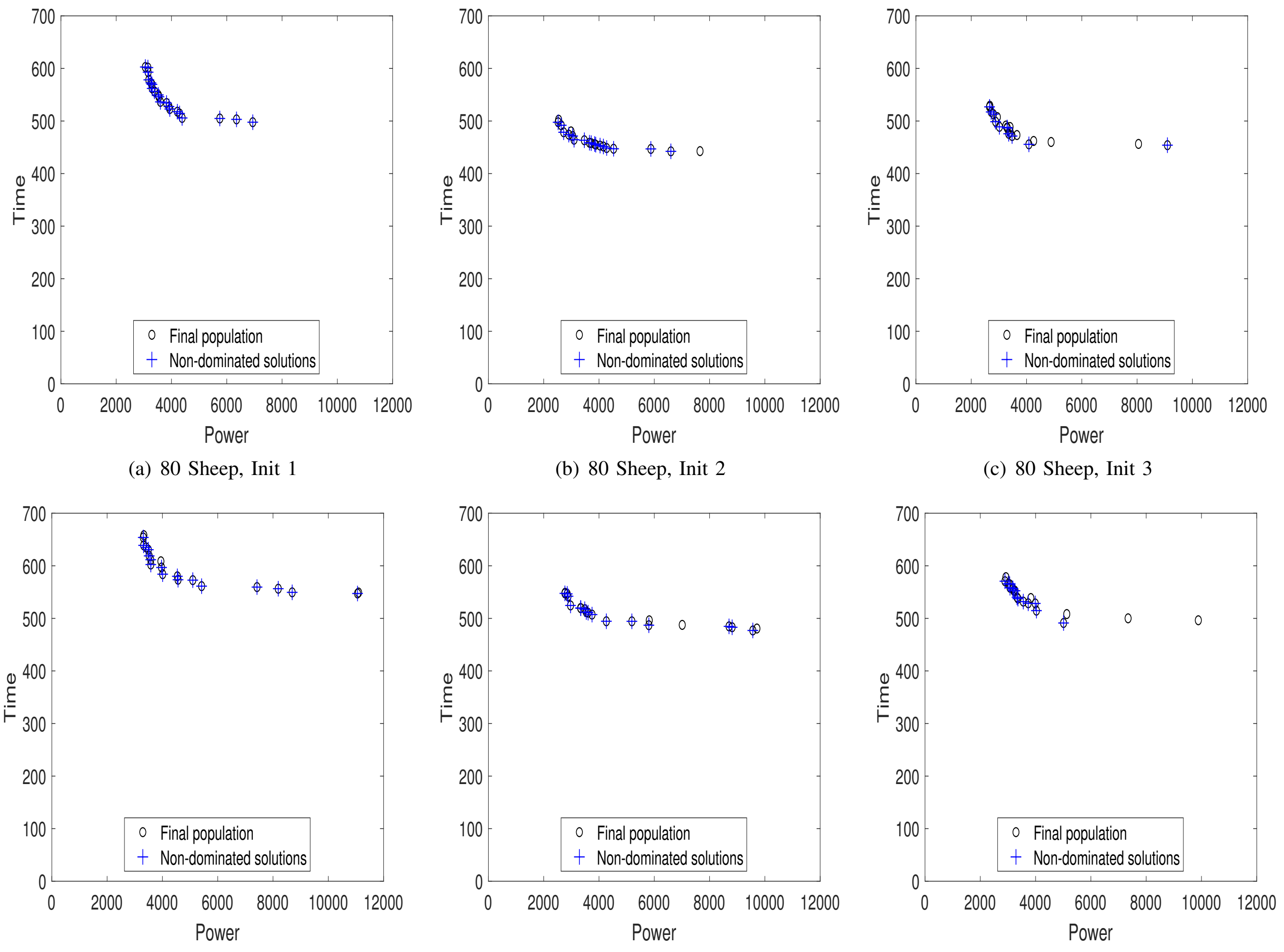
The analysis was motivated by two questions. Can we demand a minimum success rate for modulation? Is there a trade-off between power saving and the time taken to complete a task successfully? In other words, a task with effective modulations would result in a lower power, but this may take longer. We investigate both questions next, by considering a bi-objective formulation. The two objectives are (a) to minimize the time taken for task completion, and (b) to minimize power. While optimizing the two objectives, we also want to ensure that certain level of success rate is met.

We use the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [22] implementation in PlatEMO [23] to solve the above problem, with the steepness of the modulation function as the decision variable. Given the expensive nature of the simulation, we use a population size of 20 and run the problem for 500 evolutions. Simulated Binary Crossover (SBX) and Polynomial mutation were used with the default probability and distribution index set in the framework (1 and 20 respectively). To evaluate each solution, an average value of 10 simulations was used.

We demanded a minimum success rate as a hard constraint. Constraint handling in NSGA-II, also known as "parameter-less" constraint handling or "feasibility-first" constraint handling, relies on three simple ranking rules.

1) For two infeasible solutions, the one with lower constraint violation is better.
2) When comparing a feasible and infeasible solution, the feasible one is chosen regardless of objective values.
3) When comparing two feasible solutions, non-dominance and crowing distance are used to decide on an appropriate rank.

We experimented with two minimum success rates of 60% and 80% and two ranges for α_{β_j}: one where $0 \leq \alpha_{\beta_j} \leq 10$ to be consistent with the sensitivity analysis discussed above and



(a) 80 Sheep, Init 1 (b) 80 Sheep, Init 2 (c) 80 Sheep, Init 3 (d) 120 Sheep, Init 1 (e) 120 Sheep, Init 2 (f) 120 Sheep, Init 3

Fig. 2. Trade-off Solutions obtained with minimum success rate threshold set as 60% and $\alpha \in [0, 10]$

a second, $0 \leq \alpha_{\beta_j} \leq 1$, the latter sub-region was selected as our experimentation indicated that low values of α_{β_j} seemed to give better results. A factorial design of these two dimensions (success rate and α_{β_j}) created four scenarios. Each scenario was tested for the three swarm sizes of 80, 120 and 180, and the three initialization methods, creating $4 \times 3 \times 3 = 36$ sets of results.

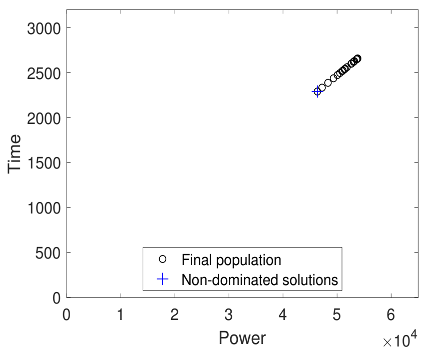
We divide the discussion in two parts. First, we discuss the cases with swarm size of 80 and 120 where the different scenarios had minor effect on the results. Second, we discuss the case with a swarm size of 180, which demonstrated variations in performance in different scenarios.

Figure 2 shows the final population in one of the runs and the obtained non-dominated front, where the minimum success rate was set to 60% and $0 \leq \alpha_{\beta_j} \leq 10$. Other corresponding cases (success rate 80% and $0 \leq \alpha_{\beta_j} \leq 1$) showed almost identical trends. The key finding from this figure is the clear trade-off that exists between power and time.

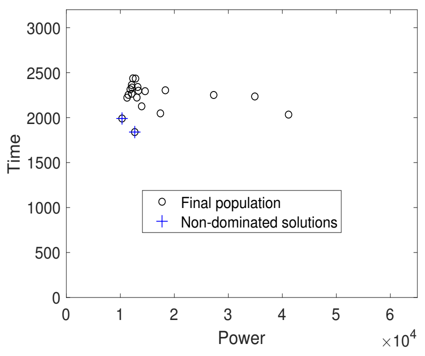
As we move from the lower swarm size scenarios (where it was easier to achieve success) to scenarios where the swarm size is larger and therefore, it was harder to succeed, Figure 3 demonstrates the effect of this increase in task complexity. We see in some of the figures that the obtained non-dominated set only had 1-3 solutions in all cases. Narrowing down the range of α_{β_j} seemed to improve convergence and clearly demonstrate that the lower α_{β_j}, the smoother the steepness of the modulation function, and therefore the better values for both power and speed. Equally interesting is that it was possible to achieve a minimum success rate of 80% in the case of the first two initialization methods, despite that this was not achievable during the sensitivity analysis stage. Thus, the utility of the bi-objective formulation and optimization is established in obtaining better solutions in terms of the three key quantities of interest, i.e., energy used, time of completion and success rate of completion of the task.

VI. Conclusion and Future Work

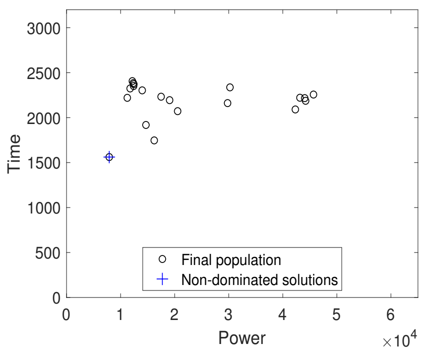
In this paper, we presented a strategy to modulate the force vector for a shepherd agent controlling a swarm of sheep. We demonstrated that as the size of the swarm increases, self-modulation of the force vector increases success rate and reduces power consumption. We explored the steepness



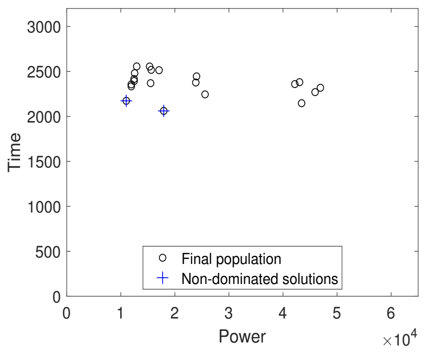
(a) 180 Sheep, Init 1 and $0 \leq \alpha \leq 10$



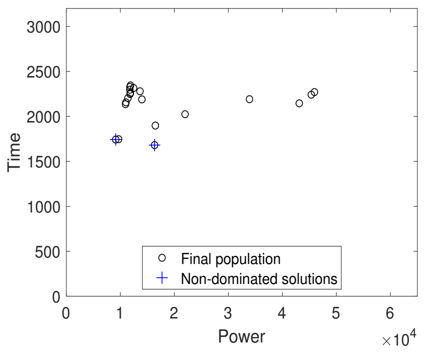
(b) 180 Sheep, Init 2 and $0 \leq \alpha \leq 10$



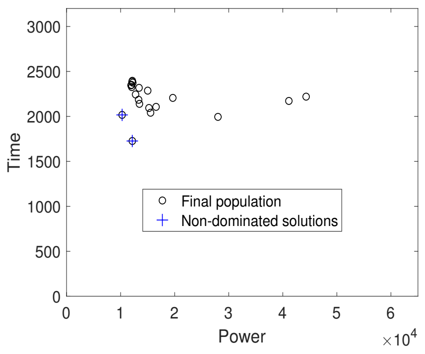
(c) 180 Sheep, Init 3 and $0 \leq \alpha \leq 10$



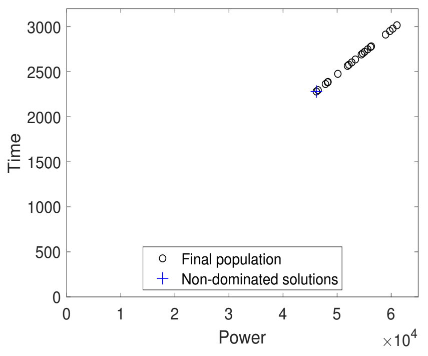
(d) 180 Sheep, Init 1 and $0 \leq \alpha \leq 1$



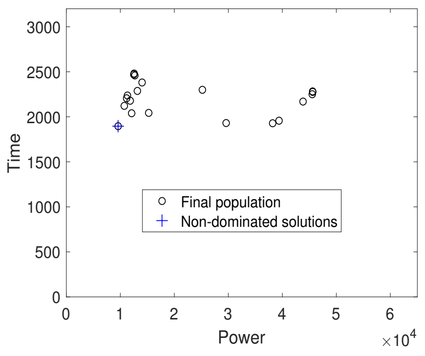
(e) 180 Sheep, Init 2 and $0 \leq \alpha \leq 1$



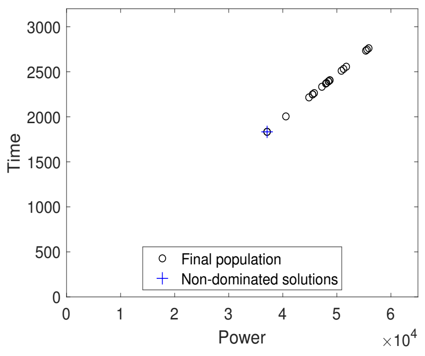
(f) 180 Sheep, Init 3 and $0 \leq \alpha \leq 1$



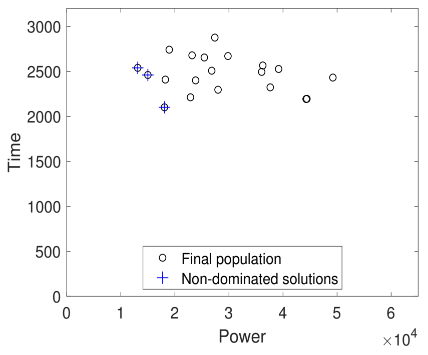
(g) 180 Sheep, Init 1 and $0 \leq \alpha \leq 10$



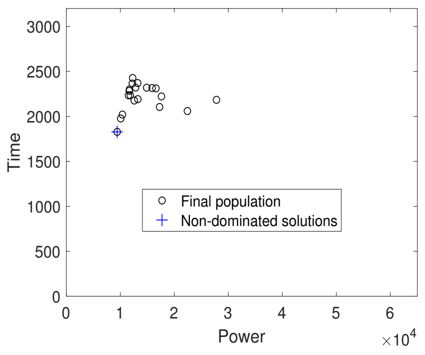
(h) 180 Sheep, Init 2 and $0 \leq \alpha \leq 10$



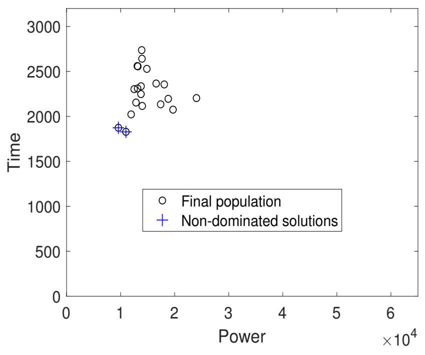
(i) 180 Sheep, Init 3 and $0 \leq \alpha \leq 10$



(j) 180 Sheep, Init 1 and $0 \leq \alpha \leq 1$



(k) 180 Sheep, Init 2 and $0 \leq \alpha \leq 1$



(l) 180 Sheep, Init 3 and $0 \leq \alpha \leq 1$

Fig. 3. Tradeoff Solutions obtained for 180 sheep with minimum success rate threshold set as 80% (upper two rows) and 60% (lower two rows) with $\alpha \in [0, 10]$ (first and third rows) and $\alpha \in [0, 1]$ (second and fourth rows)

parameter of the modulation function and identified that this parameter is critical for each form of initialization. We further used NSGA-II to identify the best value for this parameter. The results revealed that for less complex problems where success is easier to achieve, there is a trade-off between power and speed of the shepherd to complete the task. For harder problems with large swarm size, the best performance was obtained with a narrower range for α_{β_j}. Searching with lower values for the boxing constraint on α_{β_j} made it easier for NSGA II to find better spread of non-dominated sets, although convergence was still worse than that achieved when addressing the easier scenarios with swarm-size 80.

The results also revealed that smaller values for α_{β_j} created a smoother steepness of the modulation function, which led to better values for both power and speed. Moreover, it was possible to reach a higher level of success with modulation than without.

In this paper, we presented our first attempt to incorporate modulation of the force vector in shepherding. Our future work will explore this area further, varying the form of the modulation function and offering more efficient ways to optimize the steepness of the modulation function.

Acknowledgment

The authors would like to acknowledge the US Office of Naval Research - Global (ONR-G) Grant.

References

[1] K. Fujioka and S. Hayashi, "Effective shepherding behaviours using multi-agent systems," in *Region 10 Conference (TENCON), 2016 IEEE*. IEEE, 2016, pp. 3179–3182.

[2] K. Fujioka, "Effective herding in shepherding problem in v-formation control," *Transactions of the Institute of Systems, Control and Information Engineers*, vol. 31, no. 1, pp. 21–27, 2018.

[3] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, and D. JT, "Solving the shepherding problem: heuristics for herding," *Journal of The Royal Society Interface*, 2014.

[4] D. Strömbom and A. J. King, "robot collection and transport of objects: a biomimetic process," *Frontiers in Robotics and AI*, vol. 5, 2018.

[5] T. Miki and T. Nakamura, "An effective rule based shepherding algorithm by using reactive forces between individuals," *International Journal of InnovativeComputing, Information and Control*, vol. 3, no. 4, pp. 813–823, 2007.

[6] T. Miki and T. Nakamura, "An effective simple shepherding algorithm suitable for implementation to a multi-mmobile robot system," in *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*, vol. 3, Aug 2006, pp. 161–165.

[7] J.-M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *IEEE International Conference on Robotics and Automation*, vol. 4. Citeseer, 2004, pp. 4159–4164.

[8] W. Lee and D. Kim, "Autonomous shepherding behaviors of multiple target steering robots," *Sensors*, vol. 17, no. 12, p. 2729, 2017.

[9] E. Masehian and M. Royan, "Cooperative control of a multi robot flocking system for simultaneous object collection and shepherding," in *Computational Intelligence*. Springer, 2015, pp. 97–114.

[10] Y. Sueoka, M. Ishitani, and K. Osuka, "Analysis of sheepdog-type robot navigation for goal-lost-situation," *Robotics*, vol. 7, no. 2, p. 21, 2018.

[11] S. Razali, Q. Meng, and S.-H. Yang, "Immune-inspired cooperative mechanism with refined low-level behaviors for multi-robot shepherding," *International Journal of Computational Intelligence and Applications*, vol. 11, no. 01, p. 1250007, 2012.

[12] M. H. Linder and B. Nye, "Fitness, environment and input: Evolved robotic shepherding," 2010.

[13] A. Schultz, J. J. Grefenstette, and W. Adams, "Roboshepherd: Learning a complex behavior," *Robotics and Manufacturing: Recent Trends in Research and Applications*, vol. 6, pp. 763–768, 1996.

[14] B. Bat-Erdene and O. Mandakh, "Shepherding algorithm of multi-mobile robot system," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, April 2017, pp. 358–361.

[15] M. Evered, P. Burling, and M. Trotter, "An investigation of predator response in robotic herding of sheep," *International Proceedings of Chemical, Biological and Environmental Engineering*, vol. 63, pp. 49–54, 2014.

[16] Y. Tsunoda, Y. Sueoka, and K. Osuka, "On statistical analysis for shepherd guidance system," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2017, pp. 1246–1251.

[17] M. Baumann and H. K. Büning, "Learning shepherding behavior." Ph.D. dissertation, University of Paderborn, 2016.

[18] C. K. Go, B. Lao, J. Yoshimoto, and K. Ikeda, "A reinforcement learning approach to the shepherding task using sarsa," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 3833–3836.

[19] J. Brulé, K. Engel, N. Fung, and I. Julien, "Evolving shepherding behavior with genetic programming algorithms," *arXiv preprint arXiv:1603.06141*, 2016.

[20] J. Gomes, P. Mariano, and A. L. Christensen, "Cooperative coevolution of partially heterogeneous multiagent systems," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 297–305.

[21] A. Özdemir, M. Gauci, and R. Gross, "Shepherding with robots that do not compute," in *Artificial Life Conference Proceedings 14*. MIT Press, 2017, pp. 332–339.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[23] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.