

Path Planning for Shepherding a Swarm in a Cluttered Environment using Differential Evolution

Saber Elsayed*, Hemant Singh*, Essam Debie*, Anthony Perry†, Benjamin Campbell†, Robert Hunjet†, Hussein Abbass*

*School of Engineering and Information Technology, University of New South Wales, Canberra ACT, Australia.

Emails: {s.elsayed,h.singh,e.debie,h.abbass}@unsw.edu.au

†Defence Science and Technology, Australian Department of Defence, Edinburgh SA, Australia.

Emails: {Anthony.Perry,Benjamin.Campbell,Robert.Hunjet}@dst.defence.gov.au

Abstract—Shepherding involves herding a swarm of agents (*sheep*) by another a control agent (*sheepdog*) towards a goal. Multiple approaches have been documented in the literature to model this behaviour. In this paper, we present a modification to a well-known shepherding approach, and show, via simulation, that this modification improves shepherding efficacy. We then argue that given complexity arising from obstacles laden environments, path planning approaches could further enhance this model. To validate this hypothesis, we present a 2-stage evolutionary-based path planning algorithm for shepherding a swarm of agents in 2D environments. In the first stage, the algorithm attempts to find the best path for the sheepdog to move from its initial location to a strategic driving location behind the sheep. In the second stage, it calculates and optimises a path for the sheep. It does so by using *way points* on that path as the sequential sub-goals for the sheepdog to aim towards. The proposed algorithm is evaluated in obstacle laden environments via simulation with further improvements achieved.

Index Terms—Differential Evolution, Path Planning, Shepherding, Swarm Guidance.

I. INTRODUCTION

Shepherding refers to the management and guidance of a sheep herd using one or more sheepdogs. In computational intelligence research, the concept is used more broadly to model and analyze the behaviour of biologically inspired swarms, where multiple agents of different type interact with each other in a proactive and reactive manner. The reactive agents are analogous to the sheep in the problem; they respond to the presence of the proactive agent, the sheepdog, and are repulsed from it. The sheepdog makes a sequence of decisions to influence the sheep and to guide them towards a goal area. A recent comprehensive review on the subject can be found in [1]. The shepherding problem using robotic swarms is of interest in several applications beyond the biological inspiration of shepherding itself; applications include crowd control [2], clean-up of oil spills [3], disaster relief and rescue operations [4], and security/military procedures [5], among others.

The shepherding problem shares some similarities with problem of efficient navigation of mobile robots, where path

planning has been widely studied in the literature. A recent review of such methods for path planning appears in [6]. The navigation may be global or local in nature. The former assumes complete information of the environment a priori and creates an efficient path to move towards a pre-defined goal. The latter is more concerned with dynamic changes based on the relative positions among various elements. They are also referred to as reactive approaches as they consider the environment to adapt their path and are able to navigate autonomously. Correspondingly, the approaches for path planning can be broadly categorized into two classes: classical and reactive [6].

Evidently, the path planning problem becomes more difficult to solve with an increasing number of agents navigating the environment, as well as an increasing number of obstacles in the environment, among other factors. With shepherding, as opposed to a generic navigation, the problem is further complicated since the path planning algorithm needs to consider two different types of agents (sheepdog and sheep). The sheepdog, which acts as the controlling agent in the problem, needs to consider the optimisation of its own path towards a driving point, as well as clustering the sheep (noting that they behave with their own dynamics) into a flock and driving them towards the goal, while negotiating the other obstacles in the environment.

In this paper, we present an evolutionary path planning approach for shepherding that takes into account the collection and movement of the swarm (sheep) in addition to the sheepdog. The problem is different from conventional path planning for robot navigation in the sense that the control agents (sheepdog) have access to global information when seeking an optimal path, while the movement of others (sheep) is purely reactive. The two-phase algorithm starts by identifying the path for the sheepdog to move from any initial position to a position behind the swarm. The path is constrained to be obstacle free and so as not to impact the sheep; lest the sheep be repulsed and scatter, making their collection even harder and more time-consuming. In the second phase, the algorithm plans the path for the sheepdog by identifying the next series of way points to guide the sheep towards their final destination.

We identify from the related work (given in Section II) that the potential of evolutionary algorithms for solving path planning problems in shepherding has not been significantly

explored. In particular, we are interested in shepherding in environments cluttered by several obstacles. For this study, we assume that there is only one sheepdog and multiple sheep. Along with collision avoidance, the complex interplay of forces between the sheepdog and sheep makes the task even more challenging, for which the classical techniques are not geared.

We propose the use of a differential evolution (DE) based approach to path planning within the shepherding problem. The key contributions of this study are as follows:

- improvement in the baseline Strömbom model itself by considering the positions of the isolated sheep relative to the flock, in order to decide on collecting/driving behavior.
- the introduction of a two stage path planner within the shepherding problem addressing how the shepherd approaches the driving point whilst not disturbing the flock, and how it can then drive the flock to the goal,
- the ability to plan for driving using an evolutionary approach whilst taking into account obstacles that the flock may encounter (note, from a scalability perspective the path needs only to be calculated by the shepherd, not by any of the sheep), and
- validating the above contributions and the efficacy via simulation, showing that our DE path planning approach can indeed benefit shepherding.

In the remainder of this paper, we first provide a brief outline of selected works from the literature in Section II. The proposed algorithm is presented in detail in Section III, followed by experimental design and results in Section V. Concluding remarks are given in Section VI.

II. RELATED WORK

In this section, we review some of the existing work relevant to this study, focusing on three key aspects - shepherding in general, path planning and the use of computational intelligence methods in the context of these problems.

A. Shepherding

The inspiration for modelling of shepherding can be traced back to the study of animal behaviours in [7]. The development of robotic shepherd was then formalised in subsequent research such as [8], [9]. Since then, a number of works have been conducted on the topic, including the simulation and analysis of the swarming behaviours, guidance strategies, and implementation on real systems, most of which are reviewed in [1].

A shepherding problem can be initialised with a defined boundary of the field, a defined goal area to which the sheep need to be driven to, the initial position(s) of sheepdog and sheep, and the obstacles (if any) that are present in the field [10]. An obstacle refers to a region in the field that is inaccessible to the agents. Moreover, depending on the problem of interest, the behaviour of the agents, as well as their capabilities can be defined. For example, if shepherding is done through unmanned aerial vehicles (UAVs), then they

can negotiate the ground obstacles by simply flying over them instead of going around them.

Starting from the initial state, the sheepdog's aim is to group the sheep in a cohesive flock and guide them towards a goal. The primitive behaviours are defined as *collecting* and *driving* in a paper by Strömbom et al [11]. The relative movement of the sheepdog and sheep are modelled using attraction-repulsion forces, which are calculated using various weighting factors including those for collision avoidance, inertia, attraction towards the centre of mass, and natural Jittering movements.

B. Path planning

Existing path planning approaches for mobile robot navigation are classified into two types: classical and reactive [6]. We discuss these below.

1) *Classical path planning approaches*: Some examples of classical path planning approaches include:

- Cell decomposition (CD) approach: It divides the search space into a number of non-overlapping cells, with the starting and final positions assigned to specific cells [12], [13]. The path is constructed using a sequence of connected cells that do not contain an obstacle. The cells containing an obstacle could then be further split into smaller sizes to create feasible candidate cells for construction of a more efficient path. The cell shapes are often considered to be regular (e.g. grid with square cells), but a number of approaches also consider irregular shapes.
- Roadmap approach (RA): The roadmap approach resembles a graph whereby the connections (edges) between different points (nodes) can be traversed to construct the path. Voronoi diagrams and visibility graphs are some of the commonly used techniques to construct the roadmap [14], [15].
- Artificial potential field (APF) approach: In this approach, a potential field is created in the search region by considering the agents and obstacles akin to charged particles [16], [17]. The forces exerted on the robot cause attraction towards the goal and repulsion from the obstacles. A feasible path can thus be constructed via the resultant field.

2) *Reactive path planning approaches*: One of the perceived limitations of classical approaches is that they are not suitable for real-time applications involving uncertainty. The use of reactive path planning approaches has been proposed in the literature to provide more robust solutions in such cases. The reactive approaches do not assume global knowledge and instead make decisions based on local sensory data to execute in effect a 'mapless' navigation. A range of reactive methods is surveyed in [18], with a particular focus on model predictive and sliding mode control. The techniques for collision avoidance include potential field methods for moving obstacles, reciprocal methods, and hybrid logic approaches. In [6], the focus of the reactive approaches reviewed was on the use of metaheuristics to solve the underlying optimisation and modelling problems of the mobile agents. A number of such

techniques have been applied in this context, such as genetic algorithms [19]–[21], fuzzy logic [22], neural networks [23], and swarm intelligence based search methods [24]–[26].

Even though path planning has been a well-researched area in general for multi-agent robotic systems, the works focusing specifically on shepherding have been relatively few. A roadmap based approach was studied in [27] for shepherding in complex environments and shown to perform better than a bitmap (cell decomposition based) approach. In [28] the path of the sheepdog was modelled based on the current state of shepherding (approach or steer). While the former involves moving in straight lines, safe zones, or dynamic roadmaps, the latter involves positioning directly behind, side-to-side movement or turning the flock. The roadmap approach was also further extended to deal with more diverse environments in [29]. In [30], the sheep were considered collectively as deformable shapes instead of individual agents to improve scalability and robustness of the shepherding task. And finally, in [31], a circular path was enforced so that the sheepdog does not split the flock while moving to the driving position.

III. SHEPHERDING MODEL

In this section, we describe the details of the shepherding models we use for this study. Our approaches utilise the model proposed by Strömbom et al. [11], discussed in Section III-A. However, we propose an improvement in the algorithm, we call CAD-SHEEP¹, discussed in Section III-B. After demonstrating the improvements gained by this approach, we further utilise it as a baseline for demonstration of the efficiency gains which can be afforded by our DE path planning method.

A. Strömbom et al's algorithm

The Strömbom et al. [11] model, governs the dynamics of the sheepdog and sheep, by defining the specific manners in which they interact with each other and the environment. In principle, the movement of sheep is represented as a weighted linear combination of *force vectors* which represent the influence of various entities on the sheep.

The set of sheep agents are denoted by $\Pi = \{\pi_1, \dots, \pi_i, \dots, \pi_N\}$, while the shepherd (sheepdog) agents denoted as $B = \{\beta_1, \dots, \beta_j, \dots, \beta_M\}$, the set of behaviours in the simulation with $\Sigma = \{\sigma_1, \dots, \sigma_K\}$, and the set of obstacles as $O = \{O_1, O_2, \dots, O_N\}$. In our model, the obstacles induce a repulsive force on the sheep in a similar manner in that the sheep are repulsed by the sheepdog. As per [11], [32], the agents adopt different behaviors:

- 1) 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, $F_{\beta_j cd}^t$.
- 2) Collecting behavior: If one of the sheep is further away from the group, the sheepdog drives to a collection point

(with a normalized force vector $F_{\beta_j cd}^t$) behind this sheep to move it to the herd.

- 3) Sheepdog β_j adds a random force, $F_{\beta_j \epsilon}^t$, at each time-step to help resolve deadlocks. The strength of this angular noise is denoted by $W_{e\beta_j}$.
- 4) Sheepdog β_j total force $F_{\beta_j}^t$ is then calculated as:

$$F_{\beta_j}^t = F_{\beta_j cd}^t + W_{e\beta_j} F_{\beta_j \epsilon}^t \quad (1)$$

- 5) Sheep π_i is repulsed from sheepdog β_j using a force $F_{\pi_i \beta}^t$.
- 6) Sheep π_i is repulsed from other sheep $\pi_{i1}, i_1 \neq i$ using a force $F_{\pi_i \pi_{i1}}^t$.
- 7) Sheep π_i is attracted to the center of mass of its neighbors $\Lambda_{\pi_i}^t$ using a force $F_{\pi_i \Lambda_{\pi_i}^t}^t$.
- 8) Sheep π_i angular noise uses a force $F_{\pi_i \epsilon}^t$.
- 9) Sheep π_i total force is calculated as:

$$F_{\pi_i}^t = W_{\pi v} F_{\pi_i}^{t-1} + W_{\pi \Lambda} F_{\pi_i \Lambda_{\pi_i}^t}^t + W_{\pi \beta} F_{\pi_i \beta_j}^t + W_{\pi \pi} F_{\pi_i \pi_{i-1}}^t + W_{e\pi_i} F_{\pi_i \epsilon}^t \quad (2)$$

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: $S_{\beta_j}^t = 0$, otherwise it will use its default speed, $S_{\beta_j}^t = S_{\beta_j}$. The speed of a sheep is assumed constant; that is, $S_{\pi_i}^t = S_{\pi_i}$

$$P_{\pi_i}^{t+1} = P_{\pi_i}^t + S_{\pi_i}^t F_{\pi_i}^t \quad (3)$$

$$P_{\beta_j}^{t+1} = P_{\beta_j}^t + S_{\beta_j}^t F_{\beta_j}^t \quad (4)$$

Both π and β agents in Strömbom model move with fixed speed. Generally, this is neither biologically plausible, since sheepdogs, for example, do not move with a constant speed during shepherding, nor technologically appropriate considering vehicle dynamics. Constant speed also limits the ability for the model to encapsulate behavioural attributes pertaining to closing speed (e.g., aggressiveness) that can influence both the effectiveness and efficiency of shepherding [32].

B. Modified algorithm (CAD-SHEEP)

When analysing Strömbom et al's original algorithm, it was apparent that the sheepdog lacked the intuition of its biological counterpart. In particular, Figure 1 shows two sheep outside the flock range: $F1$ and $F2$. According to Strömbom et al, $F1$ is the furthest sheep, and should be collected. However, it could be inefficient to proceed to collect $F1$ since the herd will encounter $F1$ as it approaches the goal, and the natural attraction force between $F1$ and the members of the herd will provide cohesion. As such, we discount the furthest sheep if it is located between the two brown parallel lines in Figure 1. These two parallel lines are both perpendicular on the vector

¹a name formed by rearranging the first letter of surnames of all contributors to the model development in the research group

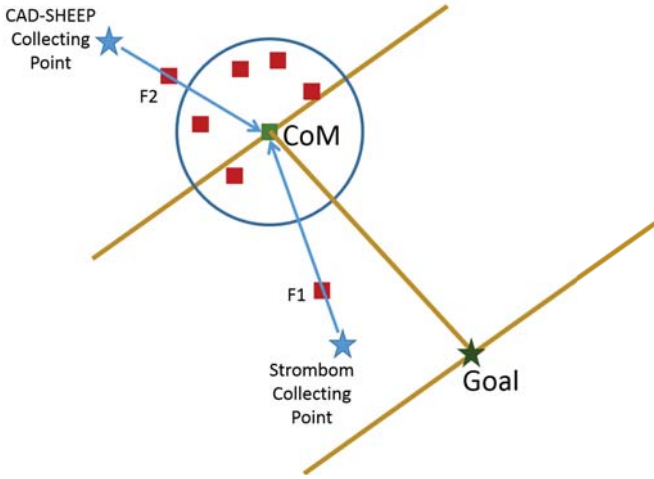


Fig. 1. The difference between Strömbom et al and CAD-SHEEP shepherding algorithm.

between the Centre of Mass (CoM) and the goal. Only sheep outside the herd zone and outside the area between these two lines are considered in the calculation of furthest sheep. In the situation depicted in Figure 1, Strömbom et al would move to collect F1, while the CAD-SHEEP algorithm will select F2.

IV. METHODOLOGY

This section discusses the framework and key components of the proposed methodology for path planning.

A. Two-Phase Framework For Sheep Driving and Collision Avoidance

Our framework offers a two-phase solution combining collision-free path planning for both the shepherd to reach the driving position and then for the flock to be driven to the designated goal. Particularly, within the first stage, Differential Evolution (DE) is used to optimise the shortest path to move the shepherd to a driving point which does not adversely affect the flock, while in the second stage, the shortest path from the flock's GCM to target position is optimised to drive the sheep whilst avoiding collision with obstacles in the environment.

Algorithm 1 Two-stage shepherding framework

```

 $P_{\beta_j} \leftarrow$  current position of the shepherd  $\beta_j$ 
 $P_{d_j} \leftarrow$  the target driving position for shepherd  $\beta_j$ 
 $A_{\beta}^* \leftarrow$  ordered list of waypoints calculated using spline
interpolation along the optimised path from  $P_{\beta_j}$  to  $P_d$ 
Move the shepherd to all points in  $A_{\beta}^*$ 
 $A_{goal}^* \leftarrow$  ordered list of sub-goals
while  $A_{goal}^*$  is not empty do
     $p_i \leftarrow$  the next way point in  $A_{goal}^*$ 
    Drive the flock to  $p_i$  with fixed speed
end

```

The proposed framework begins by determining the driving position $P_{d_j} = GCM + r_a$ behind the flock in the direction to the goal, where r_a is the shepherd influence range. Once P_{d_j} is

determined, the best possible path to move the shepherd from its current position to P_{d_j} is calculated using the proposed DE path planning algorithm, see section IV-B. For this step, each sheep is considered as an obstacle with its radius R_{pi_i} equal to the repulsion from the sheepdog, this allows a safety zone around each sheep, hence no dispersion occurs. The DE algorithm returns the best possible path A_{β}^* with D way-points calculated using spline interpolation as follows:

$$A_{\beta}^* = [w_{\beta}^1, w_{\beta}^2, \dots, w_{\beta}^d, \dots, w_{\beta}^D] \quad (5)$$

To guarantee a smooth path, spline interpolation is used to generate points between these way-points. The shepherd then traverses the way-points generated to reach its driving position (see Figure 2).

Once the shepherd arrives at the driving point, a safety zone circle C_F is generated around the flock centred at its GCM. A second optimisation cycle begins to find the best path from the flock GCM to target position whilst ensuring no obstacles are within C_F . Algorithm 1 summarises the proposed framework and such a path is depicted in Figure 2.

B. Differential Evolution Based Path Planning Algorithm

Our approach utilises Differential Evolution (DE) for optimization of the location of the way points. The process starts with a random initial population of size PS . Each solution represents a path $A = [w^1, w^2, \dots, w^d, \dots, w^D]$ with D way-points, each containing its x and y coordinates, allowing for representation as a two-dimensional array. For presentation simplicity, the DE steps discussed below consider only the x values, yet the evolutionary steps are applied to the y -coordinates as well. We initialise our possible solutions, $\vec{x}_z (\forall z = 1, 2, \dots, PS)$ within the search space, such that

$$x_{z,j} = \underline{x}_j + rand_j(0, 1) \times (\bar{x} - \underline{x}), \forall j = 1, 2, \dots, D \quad (6)$$

where $rand_j(0, 1)$ is a uniform random number within $[0, 1]$, \underline{x} and \bar{x} are the lower and upper boundaries of the search space, here set to zero and paddock size, respectively).

Each individual is evaluated based on both its objective function and adherence to constraints. At each population evaluation, the number of current fitness evaluations (cfe) is increased. Then, the entire population is evolved by DE operators (mutation and crossover); that is, for every solution \vec{x}_z , a new trial individual, \vec{u}_z is generated using:

$$u_{z,j} = \begin{cases} x_{z,j} + F_z \cdot (x_{\phi,j} - x_{z,j} + x_{r_1,j} - x_{r_2,j}) & \text{if } (rand \leq cr_z \text{ or } j = j_{rand}) \\ x_{z,j} & \text{otherwise} \end{cases} \quad (7)$$

where F_z is the amplification factor, cr_z is the crossover rate, $r_1 \neq r_2 \neq z$ are random integer numbers, with \vec{x}_{r_1} and \vec{x}_{r_2} randomly selected from X , $x_{\phi,j}$ was selected from the best 10% individuals in X [33]. Note if $u_{z,j}$ violates the search boundary, it is rounded back to the limit, i.e., if $u_{z,j} < \underline{x}$ then $u_{z,j} \leftarrow \underline{x}$, and if $u_{z,j} > \bar{x}$, then $u_{z,j} \leftarrow \bar{x}$. For a discussion on how we adapt F_z and Cr_z , please see subsection IV-D below.

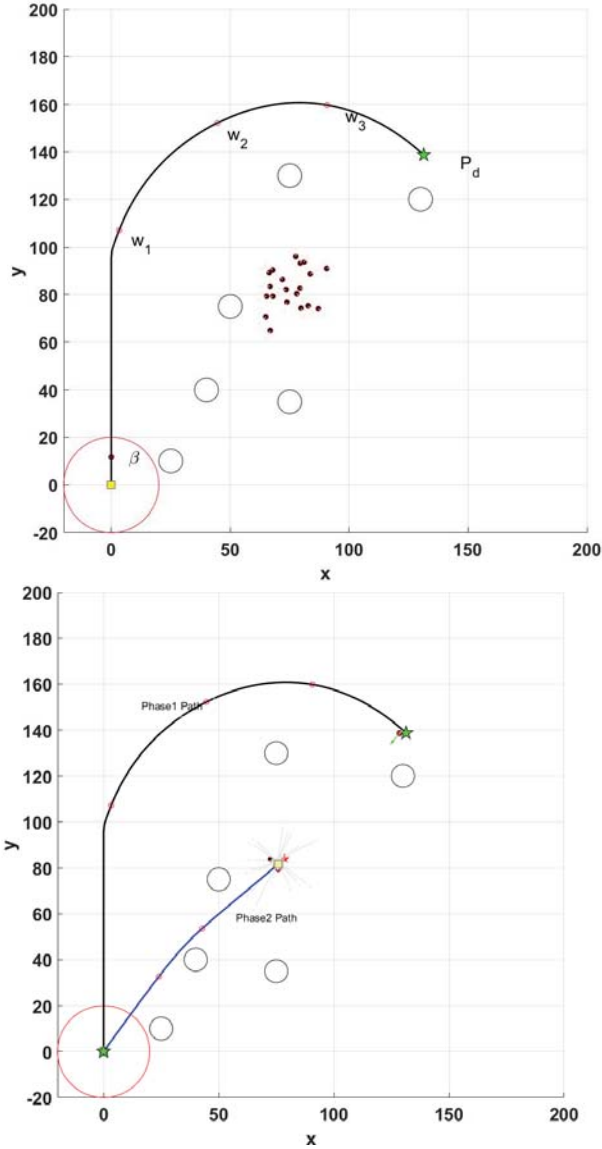


Fig. 2. **Top:** Optimising the path from sheepdog to driving point. Note that every sheep is handled as an obstacle (not shown in this figure) with a radius of 60 units. The path is constrained to be obstacle free and should not impact the sheep **Bottom:** An optimised path from the collected flock to the goal.

Once the new solutions are generated, pairwise comparison between every \vec{x}_z and \vec{u}_z is conducted, with the winner progressing to the next generation; that is \vec{u}_z survives to the next population if (1) both solutions are feasible, and $f(\vec{u}_z) \leq f(\vec{x}_z)$; or (2) \vec{u}_z is feasible, but \vec{x}_z is not; or (3) both solutions are infeasible, and $\psi(\vec{u}_z) \leq \psi(\vec{x}_z)$, where the violation of the z^{th} solution is calculated using equation 10. The algorithm continues until the stopping criterion is satisfied.

C. Evaluation

As previously mentioned, the quality of each solution is determined by its fitness value and any constraint violations.

In this paper, the objective function considered is the length of a path (L) from the start point to the target location. The following steps are carried out for every solution \vec{x}_z .

- 1) set the x -coordinate vector $XS \leftarrow [x, x_{z,1}, \dots, x_{z,D}, \bar{x}]$;
- 2) set the y -coordinate vector $YS \leftarrow [y, y_{z,1}, \dots, y_{z,D}, \bar{y}]$;
- 3) $TS \leftarrow$ split a line into $k = D + 2$ points (i.e., equal intervals);
- 4) $LS \leftarrow$ generate a vector of $p_{max} = 100$ evenly spaced points between 0 and 1;
- 5) interpolate XS and YS over unevenly-spaced sample points; i.e., generate a vector of interpolated values (\vec{XI}) corresponding to the points in LS . The values are determined by cubic spline interpolation of TS and XS . Similarly, a vector of interpolated values (\vec{YI}) corresponding to the points in LS is generated by the cubic spline interpolation of TS and YS . Note that the combination of $[\vec{XI}; \vec{YI}]$ represents the path from start to target, with each pair of XI_p and YI_p , where $p = 1, 2, \dots, p_{max}$ represents a point in this path.
- 6) Subsequently, the length (L) of this path is calculated as follows:

$$L_z = \sum_{p=1}^{p_{max}-1} \sqrt{(XI_{p+1} - XI_p)^2 + (YI_{p+1} - YI_p)^2} \quad (8)$$

Hence, the objective function is to

$$\text{minimise } L \quad (9)$$

A path is considered feasible if it does not cross any of the obstacles. Mathematically, the following equation is used to calculate the total constraint violation (how much a path overlaps with all obstacles), with a value of 0 means no overlapping exist.

- 1) calculate the Euclidean distance between each point in the path and the centre of each obstacle (O), such that $d_{p,s} = \sqrt{(XI_p - O_{x,s})^2 + (YI_p - O_{y,s})^2}$, where $O_{x,s}$ and $O_{y,s}$ are the x and y coordinates of the centre of O .
- 2) the violation of the z^{th} solutions is calculated as

$$\psi_z = \sum_{p=1}^{p_{max}} \sum_{s=1}^N \text{maximum} \left(1 - \frac{d_{p,s}}{O_{s,radius}}, 0 \right) \quad (10)$$

where $O_{s,radius}$ is the radius of the s^{th} obstacle.

During the driving behaviour, we intend to move the sheep flock considering it as a circle, so the violation is considered as

$$\psi_z = \sum_{p=1}^{p_{max}} \sum_{s=1}^N \text{maximum} \left(1 - \frac{d_{p,s} - flock_{radius}}{O_{s,radius}}, 0 \right), \quad (11)$$

where $flock_{radius}$ is the distance between the global centre of mass and the furthest sheep.

D. Adaptation of Amplification Factor and Crossover Rate

Within our implementation of DE, the technique used to adapt F and Cr considering the presence of constraints is based on the work presented in [34]; that is:

- 1) A memory archive of size H for both parameters (M_{Cr} , M_F) is set to a value of 0.5.
- 2) Every \vec{x}_z is assigned with its Cr_z and F_z

$$Cr_z = \text{randni}(M_{Cr,z}, \sigma_{cr}) \quad (12)$$

$$F_z = \text{randci}(M_{F,z}, \sigma_F) \quad (13)$$

where r_z is randomly chosen from $[1, H]$. $\text{randni}(\mu, \sigma)$ and $\text{randci}(\mu, \sigma)$ are generated using the normal and Cauchy distributions with mean μ and variance σ , $\sigma_F = \sigma_{cr} = 0.1$.

- 3) After every evolutionary iteration, successful Cr_z and F_z are recorded in S_{Cr} and S_F ; the memory archive is updated as follows:

$$M_{Cr,d} = \text{mean}_{WA}(S_{Cr}) \text{ if } S_{Cr} \neq \text{null} \quad (14)$$

$$M_{F,d} = \text{mean}_{WL}(S_F) \text{ if } S_F \neq \text{null} \quad (15)$$

where $1 \leq d \leq H$ is the position in the memory archive. Note that d is initialised to 1 and increased by 1 after every addition to the memory archive. If $d > H$, then $d = 1$. $\text{mean}_{WA}(S_{Cr})$ and $\text{mean}_{WL}(S_F)$ are calculated using the following equations:

$$\text{mean}_{WA}(S_{Cr}) = \sum_{\gamma=1}^{|S_{Cr}|} w_{\gamma} \cdot S_{Cr,\gamma} \quad (16)$$

$$\text{mean}_{WL}(S_F) = \frac{\sum_{\gamma=1}^{|S_F|} w_{\gamma} \cdot S_{F,\gamma}^2}{\sum_{\gamma=1}^{|S_F|} w_{\gamma} \cdot S_{F,\gamma}} \quad (17)$$

where

$$w_{\gamma} = \frac{\xi_{\gamma}}{\sum_{\gamma=1}^{|S_{Cr}|} \xi_{\gamma}} \quad (18)$$

and ξ_{γ} is calculated as follows:

- First, three scenarios are defined to classify solutions:
 - a) scenario-1 (**Infeasible to infeasible**): $\vec{x}_{z,t}$ and $\vec{x}_{z,t-1}$ are infeasible, where t is the current generation.
 - b) scenario-2 (**Infeasible to feasible**): $\vec{x}_{z,t-1}$ is infeasible while $\vec{x}_{z,t}$ is feasible.
 - c) scenario-3 (**Feasible to feasible**): $\vec{x}_{z,t}$ and $\vec{x}_{z,t-1}$ are both feasible.

- For each successful solution ($\gamma \in 1, 2, \dots |S_{Cr}|$) which satisfies scenario 1 conditions, the corresponding I_{γ} is

$$\xi_{\gamma} = I_{\gamma} = \max \left(0, \frac{\psi_{\gamma,t-1} - \psi_{\gamma,t}}{\psi_{\gamma,t-1}} \right) + \max \left(0, \frac{f_{\gamma,t-1} - f_{\gamma,t}}{|f_{\gamma,t-1}|} \right) \quad (19)$$

- For those solutions ($\gamma \in 1, 2, \dots |S_{Cr}|$) which belong to scenario 2 or 3, I_{γ}

$$\xi_{\gamma} = \max(0, I_{\gamma}) + \frac{\psi_{\gamma,t-1} - \psi_{\gamma,t}}{\psi_{\gamma,t-1}} + \max \left(0, \frac{f_{\gamma,t-1} - f_{\gamma,t}}{|f_{\gamma,t-1}|} \right) \quad (20)$$

V. EXPERIMENTATION AND RESULTS

A. Strömbom vs CAD-SHEEP in Obstacle Free Environment

In this subsection, we compare Strömbom et al.'s against the CAD-SHEEP variant in an obstacle free environment. Using Matlab, we create an environment of size 500x500 units, with different herd sizes comprising 10, 50 and 100 sheep. Across 10 simulation runs for each of the herd sizes, the two algorithms are compared in terms of their ability to effectively (success rate) and efficiently (speed) complete the task as shown in Table I.

The results of this experiment show that both approaches can achieve 100% success rates with different number of sheep. However, CAD-SHEEP method outperformed the standard Strömbom method in terms of number of steps taken to drive the flock to the designated goal area in all three scenarios. Particularly, with small flock sizes CAD-SHEEP can achieve up to 44% improvement on average. The results highlight also the performance stability of both methods with CAD-SHEEP having lower standard deviation (11.4 to 30% improvement over Strömbom).

The advantage of CAD-SHEEP algorithm is during collection. This advantage starts to disappear as the flock size increases. The larger flock size comes with a bias, where the total attraction force acting on sheep due to attraction to CoM is high. The flock tends to cluster more, thus, it is less likely to have a sheep requiring collection. This is evident in the case of a flock size of 100 sheep, where Strömbom et al.'s performance starts to approach that of the CAD-SHEEP algorithm.

B. Two-Phase Path Planning Framework in Cluttered Environments

Hypothesising that further improvements can be had through the use of path planning, in this subsection, we analyse the performance of the two-phase path planning framework in improving the shepherding task in cluttered environments and compare our results with the newly developed CAD-SHEEP model. Our experimental environment, implemented

² $|S_{Cr}|$ is the number of successful Cr recorded in S_{Cr} , and $|S_{Cr}| = |S_F|$

TABLE I
A COMPARISON OF STRÖMBOM ET AL AGAINST THE CAD-SHEEP
ALGORITHM IN AN OBSTACLE FREE ENVIRONMENT WITH DIFFERENT
HERD SIZE. NUMBERS REPRESENT NUMBER OF STEPS TAKEN TO ACHIEVE
MISSION.

No. of sheep	Metric	CAD-SHEEP Model	Strömbom et al. Model
10	best	370	512
	mean	410.4	732.5
	std	33.86	112.81
50	best	373	391
	mean	376.3	447.1
	std	3.49	30.61
100	best	368	372
	mean	371.6	378.9
	std	1.80	6.20

in Matlab, allowed for the comparison to be carried out across multiple scenarios. The environment size is 200x200 units; goal location for shepherding is [0,0] as is the initial shepherd location. The flock is randomly spawned with both the x and y locations are ≥ 60 and ≤ 100 . Obstacle number is either 6 or 13. Their locations are positioned in the area out the initial flock locations, and their locations are fixed over all runs of the simulation. Obstacle size is set to either small (radius=5) or large (radius=10). The flock size has three settings: 20, 40 or 80 sheep. A depiction of the different scenarios is shown in Figure 3.

The Differential evolution utilised a population size of 30 individuals with the algorithm running for 150 generations. DE's parameters, F and CR are self-adaptively updated, as per the discussion in section IV-D. The number of way-points for the path generation was set to $D = 3$. Each scenario configuration was run 20 times, with the best, average and standard deviation of the time taken (number of simulation steps) to successfully complete the shepherding task reported in Table II.

The reader will note that in all cases investigated, the addition of our two phase DE path planner greatly improves the results over the CAD-SHEEP approach. There are, however, two scenarios where a larger standard deviation was observed. We hypothesise that this variation is caused by the placement of obstacles which challenges the DE to find appropriate obstacle free paths given the low fidelity $D = 3$ way-points. Future work will conduct a sensitivity analysis on the number of way-points to allow for more complex path generation.

VI. CONCLUSION AND FUTURE WORK

The guidance of a large group of agents (a swarm) using a single point of control couples the dynamics of the control point with that of the group, and constrains the movement of the control point to that feasible for itself and the group as a whole. Within the context of shepherding, we presented a differential evolution algorithm to plan the path for the sheepdog while constraining the path to those appropriate for the sheep it herded.

TABLE II
SIMULATION RESULTS COMPARING TASK COMPLETION TIME

No. of obstacles	No. of sheep	Metric	CAD-SHEEP Model	CAD-SHEEP1 Model
6 (small)	20	best	267	185
		mean	286.95	211.65
		std	14.31	17.29
6 (small)	40	best	257	191
		mean	284.10	198.85
		std	13.30	8.40
6 (small)	80	best	256	192
		mean	339.30	199.15
		std	152.02	5.78
6 (large)	20	best	263	200
		mean	286.40	254.60
		std	12.16	105.40
6 (large)	40	best	263	196
		mean	284.80	212.90
		std	16.32	23.06
6 (large)	80	best	259	194
		mean	308.30	254.90
		std	62.51	104.93
13 (large)	20	best	214	190
		mean	259.55	215.35
		std	23.44	19.44
13 (large)	40	best	217	191
		mean	243.90	222.25
		std	16.38	37.49
13 (large)	60	best	223	190
		mean	247.45	218.75
		std	20.51	18.82

Our framework decomposed the problem into path planning to approach the flock (without inducing scattering), and then path planning, via the generation of sub-goal waypoints, to drive the flock whilst minimising encountered obstacles. The efficacy of the approach was evaluated via simulation in environments of varying flock size, obstacle number and obstacle size.

To evaluate the path planning approach, we modified the Strömbom approach to offer a more efficient algorithm for swarm guidance. The modified algorithm without path planning formed the baseline in all scenarios. This modified algorithm was then coupled with the differential evolution based path planning algorithm which provided further gains.

This study opens new interesting questions which require further work. Evaluating our proposed algorithm in different scenarios, such as multiple sheepdogs and dispersed sheep flocks will form some of our future work in this area. Moreover, relaxing the constraint that sheep are handled as obstacles with a large safety zone radius will be evaluated. A comparative study with other evolutionary algorithms would equally be useful.

ACKNOWLEDGEMENT

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

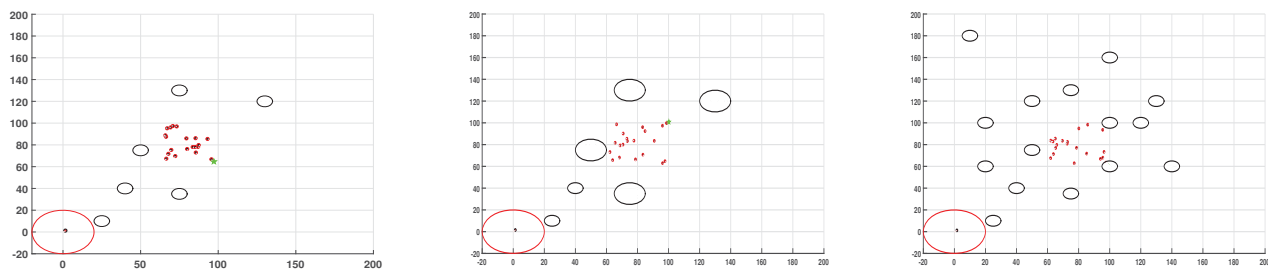


Fig. 3. Three scenario configurations with different number of obstacles and obstacle sizes

REFERENCES

- [1] N. K. Long, K. Sammut, D. Sgarioto, M. Garratt, and H. A. Abbass, "A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 523–537, 2020.
- [2] J.-M. Lien and E. Pratt, "Interactive planning for shepherd motion," in *AAAI Spring Symposium: Agents that Learn from Human Teachers*, 2009, pp. 95–102.
- [3] M. Fingas, *The basics of oil spill cleanup*. CRC press, 2012.
- [4] D. A. Shell and M. J. Mataric, "Directional audio beacon deployment: an assistive multi-robot application," in *IEEE International Conference on Robotics and Automation*, 2004, vol. 3. IEEE, 2004, pp. 2588–2594.
- [5] L. Chaimowicz and V. Kumar, "Aerial shepherds: Coordination among uavs and swarms of robots," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 243–252.
- [6] B. Patle, G. B. L. J., A. Pandey, D. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582 – 606, 2019.
- [7] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.
- [8] R. Vaughan, J. Henderson, and N. Sumpter, "Introducing the robot sheepdog project," in *Proceedings of the International Workshop on Robotics and Automated Machinery for BioProductions*, 1997.
- [9] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, "Experiments in automatic flock control," *Robotics and autonomous systems*, vol. 31, no. 1-2, pp. 109–117, 2000.
- [10] H. El-Fiqi, B. Campbell, S. Elsayed, A. Perry, H. K. Singh, R. Hunjet, and H. Abbass, "A preliminary study towards an improved shepherding model," in *Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 75–76.
- [11] 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.
- [12] M. Šeda, "Roadmap methods vs. cell decomposition in robot motion planning," in *international conference on signal processing, robotics and automation*. World Scientific and Engineering Academy and Society, 2007, pp. 127–132.
- [13] D. Glavaški, M. Volf, and M. Bonkovic, "Robot motion planning using exact cell decomposition and potential field methods," in *Proceedings of the 9th WSEAS international conference on Simulation, modelling and optimization*. World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 126–131.
- [14] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path," *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, pp. 58–66, 2008.
- [15] R. Wein, J. P. Van den Berg, and D. Halperin, "The visibility-voronoi complex and its applications," *Computational Geometry*, vol. 36, no. 1, pp. 66–87, 2007.
- [16] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.
- [17] L. Huang, "Velocity planning for a mobile robot to track a moving target à a potential field approach," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 55 – 63, 2009.
- [18] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.
- [19] M.-T. Shing and G. B. Parker, "Genetic algorithms for the development of real-time multi-heuristic search strategies," in *ICGA*. Citeseer, 1993, pp. 565–572.
- [20] P. Shi and Y. Cui, "Dynamic path planning for mobile robot based on genetic algorithm in unknown environment," in *Chinese control and decision conference*. IEEE, 2010, pp. 4325–4329.
- [21] X. Kang, Y. Yue, D. Li, and C. Maple, "Genetic algorithm based solution to dead-end problems in robot navigation," *International Journal of Computer Applications in Technology*, vol. 41, no. 3-4, pp. 177–184, 2011.
- [22] G. Castellano, G. Attolico, and A. Distanto, "Automatic generation of fuzzy rules for reactive robot controllers," *Robotics and Autonomous Systems*, vol. 22, no. 2, pp. 133–149, 1997.
- [23] Y.-K. Na and S.-Y. Oh, "Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification," *Autonomous Robots*, vol. 15, no. 2, pp. 193–206, 2003.
- [24] X.-l. Tang, L.-m. Li, and B.-j. Jiang, "Mobile robot slam method based on multi-agent particle swarm optimized particle filter," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 6, pp. 78–86, 2014.
- [25] J. Liu, S. Anavatti, M. Garratt, and H. Abbass, "Comprehensive learning particle swarm optimisation with limited local search for uav path planning," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 2287–2294.
- [26] Y. Zhang, D.-W. Gong, and J.-H. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172–185, 2013.
- [27] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Roadmap-based flocking for complex environments," in *10th Pacific Conference on Computer Graphics and Applications*. IEEE, 2002, pp. 104–113.
- [28] 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, 2004. Proceedings*, vol. 4. IEEE, 2004, pp. 4159–4164.
- [29] C. Vo, J. F. Harrison, and J.-M. Lien, "Behavior-based motion planning for group control," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3768–3773.
- [30] J. F. Harrison, C. Vo, and J.-M. Lien, "Scalable and robust shepherding via deformable shapes," in *International Conference on Motion in Games*. Springer, 2010, pp. 218–229.
- [31] N. Long, M. Garratt, D. Sgarioto, K. Sammut, and H. Abbass, "Shepherding autonomous goal-focused swarms in unknown environments using space-filling paths." Springer, accepted for publication.
- [32] H. Singh, B. Campbell, S. Elsayed, A. Perry, R. Hunjet, and H. Abbass, "Modulation of force vectors for effective shepherding of a swarm: A bi-objective approach," in *IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 2941–2948.
- [33] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.
- [34] S. Elsayed, R. Sarker, and C. C. Coello, "Enhanced multi-operator differential evolution for constrained optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 4191–4198.