Improving Wildlife Monitoring using a Multi-criteria Cooperative Target Observation Approach

Sai Krishna Munnangi
IIIT Hyderabad
krishna.munnangi@research.iiit.ac.in

Praveen Paruchuri
IIIT Hyderabad
praveen.p@iiit.ac.in

Abstract

Monitoring of wildlife is very important for maintaining sustainability of environment. In this paper, we pose Wildlife Monitoring as a Cooperative Target Observation (CTO) problem and propose a Multi Criteria Decision Analysis (MCDA) based algorithm named MCDA-CTO, to maximize the observation of different animal species by Unmanned Aerial Vehicles (UAVs) and to effectively handle multiple target types and the multiple criteria that arise due to targets and environmental factors, during decision making. UAVs have uncertainty in observation of targets which makes it challenging to develop a high-quality monitoring strategy. We therefore develop monitoring techniques that explicitly take actions to improve belief about the true type of targets being observed. In wildlife monitoring, it is often reasonable to assume that the observers may themselves be a subject of observation by unknown adversaries (poachers). Randomizing the observers actions can therefore help to make the target observation strategy less predictable. We then provide experimental validation which shows that the techniques we develop provide a higher (true positive/true negative) ratio along with better randomization than state of the art approaches.

1. Introduction

With the rapidly declining populations of wildlife, sustainable wildlife management is an important concern across the world. It is estimated that humanity may have wiped out 60% of animal populations since 1970 with dozens of species going extinct on everyday basis [1]. Wildlife monitoring is therefore a critical activity to sustain the populations which involves tracking of animal movement patterns, habitat utilization, population demographics, identifying snaring and poaching incidents and breakouts [2]. Manual monitoring may not always be possible due to the vast areas involved, different types of terrain and distraction to animals. In recent times, many wildlife conservation organizations started deploying Unmanned Aerial Vehicles (UAVs) to monitor wildlife [3, 4, 5]. UAVs can monitor large areas at a time, transmit the live feed, handle adverse climatic conditions and can even operate at night without distracting animals. Due to these significant advantages, UAVs are becoming a viable option for wildlife monitoring. Tasks involving UAVs such as species counting, surveying a region and others, typically involve a human to operate the UAVs. In this paper, we aim for better automation of wildlife monitoring using a technique named Multi Criteria Decision Aiding.

Our first step towards this, is to model wildlife monitoring as a Cooperative Target Observation problem. **Cooperative Target Observation (CTO)** refers to a general class of problems where a set of entities called **Observers** collectively try to observe another set of entities called **Targets**. The CTO problem has received attention in both multi-agents [6, 7, 8, 9] and robotics literature [10, 11, 12, 13] due to its ability to model a number of applications. Depending upon the problem specifics, the observers act independently or cooperate among themselves for maximizing the target observation. The target entities can be heterogeneous with different levels of importance and behaviors. Existing works on CTO problems focus on homogeneous target types and single criterion based decision making. We propose a Multi Criteria Decision Analysis (MCDA) based algorithm named **MCDA-CTO**, to maximize the observation of targets and to effectively handle multiple target types and the multiple criteria such as altitude when observers are modeled as drones.

2. Literature Review

Different variations of the CTO problem have been studied in literature depending on the need of the domain [14, 15, 16, 17]. The CTO problem as defined by [16] is a tuple $<S,\ O,\ X>$ where S is a two-dimensional,

URI: https://hdl.handle.net/10125/64380
978-0-9981331-3-3

bounded, enclosed spatial region, O is a team of k_1 observer robots with observation sensors of limited range (denoted by $sensor\_range$) and X is a set of k_2 targets. Given this notation, let $A(t)$ be a $k_1 \times k_2$ matrix where, $a_{ij}(t)$ is 1 if robot o_i is monitoring target x_j at time t and is 0 otherwise. The logical OR operator over a vector H is defined as,

$$\bigvee_{i=1}^{k} h_i = \begin{cases} 1 & \text{if there exists an } i \text{ such that } h_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

The goal of the problem is to maximize

$$\sum_{t=0}^{T} \sum_{j=1}^{k_2} \bigvee_{i=1}^{k_1} a_{ij}(t), \text{ given } \bigcup_{o_i \in O} sensor\_range(o_i) \ll S$$

In [14], the CTO problem is defined to allow observers to cooperate to maximize the number of targets being observed in an area of interest. [15] presents an algorithm which proposes control law for each observer robot considering the densities of observers and targets. [16] introduces Cooperative Multi-Robot Observation of Multiple Moving Targets (CMOMMT) formalism similar to CTO and develops online distributed control strategies that allow the robot team to attempt to minimize the total time in which targets escape observation. [17] studies a variant of the CTO problem in which observers have global knowledge of all the other observers and targets. The paper presents centralized, partially-decentralized and fully decentralized observer strategies using Mean Point (referred to as k-means in the paper) and Hill Climbing algorithms and shows that mean point algorithm is robust to the degree of decentralization. [18] builds upon the mean point algorithm algorithm presented by [17] to develop multiple strategies for observers in a CTO problem who have local knowledge. The paper also presents an adjustable randomization algorithm for observers named BRLP-CTO for surveillance applications.

The CTO problem modeled in our work is a decentralized version, where each observer acts individually trying to maximize its reward. There is no communication among the observers and when multiple observers simultaneously observe one target, only one of them gets the reward. Across the different variations for the CTO problem proposed in literature, the objective function is to maximize performance of the observers under various environmental settings. Different solutions have been proposed for each setting, with no universal solution to the problem.

Limitations of existing techniques: Prior works consider targets to be homogeneous agents with uniform behaviors and the decision making process to be a single criterion based on the position of targets. The assumption that all targets are of same type might not be true for all applications. In real world, targets can often be heterogeneous with different type of targets having a different level of importance. For example in disaster response, children maybe higher priority targets compared to adults who may in turn be of higher priority over animals. Existing works cannot handle heterogeneity and do not arrive at a decision based on reasoning over target types. Along with reasoning about multiple target types, observers would often need to arrive at a decision based on other environmental factors, such as battery power, distance from command center etc. Existing works are limited in these aspects and cannot be modified effectively to handle these issues. We use the BRLP-CTO algorithm presented in [18] as a benchmark for performance comparison and hence modify it appropriately to encode the wildlife monitoring domain. While BRLP-CTO cannot handle multiple criteria in general, we tailored the algorithm to incorporate multiple target types.

3. BRLP-CTO: Handling multiple target types

The BRLP-CTO algorithm presented in [18] comprises of two steps: (a) Destination D_i for the observer i is computed using mean point algorithm based on the positions of the targets in the observers sensor range. (b) Using a linear program a new destination D_i' which maximizes entropy while maintaining threshold amount of reward is computed, from the previously calculated destination D_i. The mean point algorithm of BRLP-CTO algorithm is derived from the unsupervised clustering algorithm **K-means**, that clusters N data points into K clusters. In the K-means algorithm, the K cluster centers are randomly generated first, and each data point is assigned to the cluster closest to it. After the assignment, the cluster center is updated to the mean position of data points in its cluster. Considering all the targets within the sensor range of an observer as a single cluster, the mean point algorithm calculates the mean position M_i of all the targets in observer's sensor range and updates the observer's destination position D_i as,

$$D_i \leftarrow (1 - \alpha)P_i + \alpha M_i \tag{1}$$

where P_i is the current position of observer i and α is the weighing factor. To handle multiple target types, we replace the computation of mean position M_i with

weighted mean position WM_i, which is calculated as:

$$WM_i = \sum_{j=1}^{l} \omega_{t_j} \times M_{t_j} \quad (2)$$

where M_{t_j} is the mean position of targets of type t_j within the observer i's sensor range, l is the number of different target types in the sensor range of the observer and ω_{t_j} is the weight associated with the target type t_j. The weight (ω_{t_j}) for each target type t_j is calculated as,

$$\omega_{t_j} = \frac{R_{t_j} \times n_{t_j}}{\sum_{k=1}^{l} R_{t_k} \times n_{t_k}} \quad (3)$$

where R_{t_j} is the reward obtained for observing a target of type t_j while n_{t_j} is the number of targets of type t_j in the observer's sensor range. This equation weighs the reward obtained due to a target with the number of targets of that type observed and normalizes to compute the weight for that target type.

3.1. Example

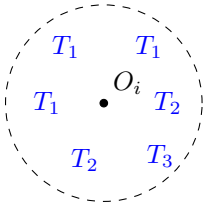

Figure 1. An example scenario

Consider an example in which there are 3 different target types $\langle T_1, T_2, T_3 \rangle$ with rewards $\langle 1, 2, 3 \rangle$ respectively. Assume that there are 3 T_1, 2 T_2 and 1 T_3 type targets in the sensor range of an observer i. Let the positions for these targets be $Po_1, Po_2, Po_3, Po_4, Po_5$, and Po_6 respectively and let M_1, M_2 and M_3 represent the mean positions of each target type. The observer would then calculate weights for each target type as,

$$\omega_{t_1} = \frac{R_{t_1} \times n_{t_1}}{\sum_{i=1}^{3} R_{t_i} \times n_{t_i}} = \frac{1 \times 3}{1 \times 3 + 2 \times 2 + 3 \times 1} = 0.3$$

Similarly, ω_{t_2} = 0.4 and ω_{t_3} = 0.3. The observer would then calculate the weighted mean position WM_i through equation 2 and compute the destination by plugging WM_i in equation 1 (note that M_i is replaced with WM_i).

3.2. Limitations of the extension

While we extended the mean point algorithm to handle multiple target types, there is a single mean point computed (for all the targets within sensor range) in the model using the equation 2 that does not allow for combinatorial reasoning over clusters of targets (8 combinations possible if 3 targets present), e.g., should the observer consider high valued targets or only the low valued ones? It also doesn't allow to make decisions optimizing other criteria e.g., with the remaining 10% battery power should the observer go to recharge point or continue pursuit of the targets?

4. The MCDA-CTO Algorithm

To handle the multiple criteria that arise due to combinatorial reasoning over multiple target types and environmental factors, we develop the MCDA-CTO algorithm based on the MCDA Clustering algorithm presented by [19]. MCDA refers to **Multiple Criteria Decision Analysis**, a technique that helps to evaluate the multiple conflicting criteria a decision maker may have. In particular, the decision maker (DM) has a set of m criteria $\{h_1, h_2, \ldots, h_m\}$ using which each possible decision (action) is evaluated and the final decision is reached. We now present the MCDA-CTO algorithm, Algorithm 1, and demonstrate it's working for the CTO problem in subsequent sections. MCDA-CTO comprises of four steps, summarized as follows: **Step 1**: Generating actions, **Step 2**: Clustering actions, **Step 3**: Evaluating actions and clusters, **Step 4**: Decision making.

4.1. Generating Actions

Actions are possible decisions that the DM can make. Depending on the problem, actions can be finite or infinite in number and they can either be generated or sampled from the solution space. The scale of measurement for each criteria can be different. An action set $A = \{a_1, a_2, \ldots, a_n\}$ consisting of possible decisions is created in this step.

4.2. Clustering Actions

This is the key idea behind the MCDA clustering algorithm [19] which we reproduce here. This step begins with initializing τ clusters and randomly assigning each action to one of the clusters. **Profiles** are created for each action in A and using a **voting procedure** the cluster profiles are created next. Using a multi-criteria **distance metric**, each action is then reassigned to the nearest cluster. Next, the profile of each cluster is updated by means of the voting procedure. The last two steps are repeated until the cluster membership no longer changes resulting in a homogeneous partition of τ clusters.

Profile: The preference structure $(P,\ I,\ J)$ representing Preference (P), Indifference (I) and Incomparability (J), captures the order of preference

Algorithm 1 MCDA-CTO Algorithm

Input: τ desired number of clusters, preference structure (P, I, J) and the evaluation function ef

Generate possible actions and create the set A.
Create Profile $Pr(a_l)$ for every action $a_l \in A$.
Initialize τ clusters $(C_1, C_2, \ldots, C_\tau)$ and randomly allocate each action to one of the clusters.
For each cluster C_j compute the profile of cluster $Pr(C_j)$.
repeat
    for all actions $a_l \in A$ **do**
        Assign a_l to the nearest cluster C_i, i.e.
        $d(a_l, C_i) \leq d(a_l, C_j)\ i, j \in \{1, \ldots, \tau\}$
    end for
    for all clusters C_j, where $j \in \{1, \ldots, \tau\}$ **do**
        Update the profile of cluster center $Pr(C_j)$.
    end for
until the cluster membership no longer changes
For each $a \in A$, calculate expected reward through ef.
Pick an action $a' \in A$ to execute through one of the four notions.

return a'

among the criteria and is defined by the DM. The relations are derived as a result of comparison of two actions a_i and $a_j \in A$ as follows:

- $a_i P a_j$ if a_i is preferred to a_j,
- $a_i I a_j$ if a_i is indifferent to a_j,
- $a_i J a_j$ if a_i is incomparable to a_j.

The notion of profile for each action is introduced using P, I, J, to determine the potential similarity between actions. The profile $Pr(a_i)$ of an action $a_i \in A$, is defined as the vector $\langle Pr_1(a_i), Pr_2(a_i), Pr_3(a_i), Pr_4(a_i)\rangle$, where:

- $Pr_1(a_i) = \{a_j \in A \mid a_i P a_j\}$
- $Pr_2(a_i) = \{a_j \in A \mid a_j P a_i\}$
- $Pr_3(a_i) = \{a_j \in A \mid a_i I a_j\}$
- $Pr_4(a_i) = \{a_j \in A \mid a_i J a_j\}$

Distance: If n is the total number of actions in A, then the distance between a_i and a_j is defined as:

$$d(i,j) = 1 - \frac{\sum_{k=1}^{4} |Pr_k(a_i) \cap Pr_k(a_j)|}{n}$$

Voting Procedure: The profile for a cluster is created by placing each action into one of the four sets with the condition that the action appears in the same set for a majority of the actions of that cluster. That is,

$$a_j \in Pr_l(C_i) \iff l = \underset{t \in 1..4}{argmax} |\{a_j \in Pr_t(a_k)\}|$$

$$\forall a_k \in C_i, \forall a_j \in A \; and \; i \in \{1, 2, \ldots, \tau\}$$

The output clusters are characterized on the basis of similarity among actions in terms of reward, resulting in actions of each cluster having similar outcome. The DM would then need to choose the cluster that optimizes his objectives and execute an action from the chosen cluster.

4.3. Evaluating Actions

Action evaluation is a step in the process to identify the Best Cluster among τ clusters. The DM has an **evaluation function (*ef*)**, that calculates the expected reward of an input action. To identify the Best Cluster, the expected rewards for all the actions in each cluster are calculated and aggregated. Next the average expected reward per action of each cluster is calculated by dividing the aggregate expected reward with size of that cluster. Best Cluster would be the cluster whose average expected reward per is the highest.

4.4. Decision Making

To randomize the final actions, we define the following 4 notions of decision making: (a) **Best Cluster, Random Action** (RABC): an action is randomly picked from the Best Cluster, (b) **Best Cluster, Best Action** (BABC): the action with highest expected reward from the Best Cluster is identified and chosen, (c) **Quantal Response Model** (QR): an action is chosen from the Best Cluster following the probability distribution $\frac{e^{r_i}}{\sum e^{r_i}}$ calculated for every action i in best cluster with expected reward r_i, (d) **Best Action of All** (BAA): the action with highest expected reward is chosen. An action is chosen using one of the four notions and is executed.

5. Wildlife Monitoring as a CTO problem

We now model wildlife monitoring as a CTO problem, where the goal is to improve observation of targets (i.e., different species of animals) despite having a limited number of observers i.e., UAVs and demonstrate the working of MCDA-CTO algorithm. While MCDA-CTO is adaptable to variety of criteria, we present the algorithm for using altitude as a criterion along with modeling each target in sensor range as a separate criterion. Targets here are animals of different species, which are in different numbers and have different reward values e.g., monitoring an endangered species can result in higher reward than a common species while the number of endangered species can be much lower. We assume observers to have information about the possible target types they may encounter during monitoring and the rewards they obtain.

Unlike ground based observers, UAVs have an additional degree of freedom namely altitude. We model this into the CTO problem by allowing the observers to alter their altitude between certain limits. In particular, we assume that there are H different altitude levels that the UAVs can make observations from and the sensor range (i.e., radius of circle with UAV as the center) increases with increase in altitude. However, the key issue here is that the observers (UAVs) have uncertain observations i.e. if the observer spots an animal of species c, it may identify that the animal belongs to c with a probability $p \leq 1$ i.e. the observer will have a probability distribution over the type of species the animal may belong to. Each altitude level has a specific **observation uncertainty** associated with it which is proportional to the altitude of the observer. We set uncertainty as 10% (i.e. p set to 0.9) at the lowest altitude level and as 50% at the highest. Hence, an observer at the lowest altitude would have a **belief state** of $\langle 0.9, 0.1 \rangle$ for target types $\langle c, \neg c \rangle$ where $\neg c$ refers to rest of target types i.e., any target type other than c.

To handle the problem using MCDA-CTO, each target within sensor range of an observer (UAV) is modeled as a criterion for the following reason: Observers can have different beliefs about two targets of the same type due to uncertainty. Modeling as separate criteria facilitates the observer to reason about different target combinations and compute the weights separately for each target. Altitude is an additional criteria, as the observer needs to optimize among different altitude levels.

5.1. Generating Actions:

The observer first generates multiple vectors (proportional to number of targets in sensor range), each of size equal to the number of targets within the sensor range. Each element of a vector has a value between 0.0 and 1.0 corresponding to the weight associated to a particular target, hence the vectors are referred to as weight vectors. In each weight vector, sum of all the elements is 1.0. Change in weight of one target will force change in the weight of other target(s) due to the constraint that sum should be 1.0, hence the vectors are Pareto optimal. An action is defined as a weight vector appended with **altitude level**, hence referred to as action vector. Size of action vector is one more than the number of targets within the sensor range of observer. Altitude level varies between 0.0 (lowest altitude) and 1.0 (highest altitude). For experimentation, we assume that the observer has only 4 altitude levels, restricting the values of last element of the vector to one of $\{0.0, 0.33, 0.66, 1.0\}$ e.g., if an observer has 3 targets in sensor range (say t_2, t_5, and t_7), then one action vector could be $\langle 0.25, 0.5, 0.25, 0.33 \rangle$ corresponding to weight vector $\langle 0.25, 0.5, 0.25 \rangle$ and altitude of 0.33. Hence, there will be three additional action vectors corresponding to the other altitude levels for the same weight vector. We also assume that the observer is aware of the amount of uncertainty present at each altitude level. Action vectors are then clustered into τ clusters, either by using ELECTRE [20] or PROMETHEE [21] method (preference structure) for profile creation.

5.2. Evaluating Actions:

There can be different evaluation functions (ef) that provide optimal solution while satisfying all the criteria, hence the choice of picking one is left to the DM. Since the observers in our case are trying to maximize observation over a weighted function of the true type of targets, we choose the ef as follows: Let cH denote the current altitude of the observer o and aH denote altitude of the action being evaluated. Let c_o^{cH} **denote the certainty in observation** at the current altitude level and u_o^{cH} **denote the uncertainty**, where $c_o^{cH} + u_o^{cH} = 1.0$. The evaluation function ***ef*** is given by,

$$ef = \begin{cases} ER_1 + P_1 + P_2, & \text{for } aH \geq cH \\ ER_2 + P_1 + P_2, & \text{otherwise} \end{cases} \tag{4}$$

where ER_1, ER_2 are the expected reward and P_1, P_2 are the penalties. The expected rewards ER_1 and ER_2 are given by,

$$ER_1 = \sum_i \omega_i \times (c_o^{aH} \times R_{t_{a_i}} + u_o^{aH} \times R_{t_{b_i}}) \tag{5}$$

$$ER_2 = \sum_i \omega_i \times [c_o^{aH} \times (c_o^{cH} \times R_{t_{a_i}} + u_o^{cH} \times R_{t_{b_i}}) + u_o^{aH} \times (u_o^{cH} \times R_{t_{a_i}} + c_o^{cH} \times R_{t_{b_i}})] \tag{6}$$

where, for each target i in the sensor range of o, t_{a_i} and t_{b_i} are the possible target types that the observer classifies target i as, ω_i is the weight associated with target i in that action and R_t denotes the reward for observing target of type t.

An observer at a higher altitude level believing t_{a_i} to be the true type of target i based on the observation $\langle t_{a_i}, t_{b_i} \rangle$ of the target, may realize its true type to be t_{b_i}, after moving to lower altitude levels where it typically has higher accuracy observations. Hence, when the observer is evaluating lower altitude level actions, it accounts for this possibility by calculating both the expected rewards, i.e. $(c_o^{cH} \times R_{t_{a_i}} + u_o^{cH} \times R_{t_{b_i}})$, $(u_o^{cH} \times R_{t_{a_i}} + c_o^{cH} \times R_{t_{b_i}})$ and combines them

using the probability with which each of them can be true. However in the case where the observer is evaluating higher altitude level actions, since it knows the true target type with greater certainty, it calculates the expected reward based on its current beliefs. The penalties P_1 and P_2 are given by:

- $\mathbf{P_1}$: $|cH - aH| \times (-3)$ [To discourage the observer from changing its altitude frequently.]
- $\mathbf{P_2}$: $aH \times (-3)$ [To discourage the observer from staying at higher altitudes continuously and enable the observer to know better regarding the ground truth of the target it is observing.]

Evaluation function ***ef*** which is the sum of expected reward of each target and the penalties, is used by the observer to calculate the expected reward of each action, based on which the observer makes it's decision. We provide results for the evaluation function and its variants in subsequent section.

5.3. Decision Making:

Once the expected reward is calculated for each action (action vector), the Best Cluster is identified and the observer picks an action to execute using one of the four notions described earlier. For the chosen action, the destination of the observer is computed as WM_o using equation 2. The entries of the chosen action vector (except the last entry) act as ω's in computation of WM_o. The observers destination altitude is set as the last entry of the chosen action vector. Action selection using MCDA-CTO can result in higher randomness than the BRLP-CTO algorithm [which explicitly introduces randomness into the mean point algorithm procedure for CTO] from the perspective of an adversary trying to learn the strategy of the observer due to: (a) Randomness in cluster generation (b) Randomness in picking the Best Cluster and (c) Randomness in picking the action from the cluster.

Scoring Methods: We calculate four different scores to measure the performance of observers across the two algorithms.

1. **Actual Score:** This score is the sum of rewards for every target that is in the sensor range of the observer. It is awarded by the system post the surveillance operation, irrespective of the beliefs the observer has about the target types. Essentially this is the ground truth score.
2. **Expected Score:** This is the score the observer calculates based on its belief of the target types. If $\langle p, (1-p) \rangle$ are the beliefs about $\langle t_a, t_b \rangle$, then the expected score is, $p \times R_{t_a} + (1-p) \times R_{t_b}$.
3. **True Negative Score:** This score represents the mis-belief the observer has about the target type (i.e. classification as false target type). $(1-p) \times R_{t_b}$ in the expected score formula is the True Negative component.
4. **True Positive Score:** This score represents the belief the observer has about the target type (i.e. classification as true target type). $p \times R_{t_a}$ in the expected score formula is True Positive component.

6. BRLP-CTO: Handling observation uncertainty

In section 3, we modified BRLP-CTO to handle multiple target types but there is no simple way to handle criteria other than targets e.g., altitude related decisions when UAVs are observers. While determining the best altitude using BRLP-CTO maybe a challenge, we can still perform reasoning about observational uncertainty which is an indicator for altitude. We now present the Stochastic belief update model to handle observation uncertainty.

Stochastic belief update model: The observer accounts for uncertain observations in its equation and calculates the expected reward, which is used to calculate the weight for each target. If n is the total number of targets in sensor range of the observer, weight ω_j for each target j is calculated as:

$$\omega_j = \frac{ER_j}{\sum_{k=1}^{n} ER_k} \tag{7}$$

where $ER_j = p \times R_{t_j} + (1-p) \times R_{t_q}$.

ER_j is the Expected Reward for each target j and $\langle p, (1-p) \rangle$ is the presented uncertainty between the target types $\langle t_j, t_q \rangle$. The weights calculated from equation 7 would then be used in equation 2 to calculate the weighted mean of the targets. Note that M_{t_j} in equation 2 would then correspond to the position of a particular target of type t_j and l corresponds to total number of targets. The observer i's destination position is updated as below, where each time a new value for α is given by BRLP-CTO.

$$P_i \leftarrow (1-\alpha)P_i + \alpha WM_i \tag{8}$$

6.1. Example continued

In the example presented in section 3.1, let observer i be a UAV operating at an altitude h from where the target can be identified correctly with a 60% probability. Let the beliefs for each of the 6 targets be:

Table 1. Parameters used in Markov Chain model

Target Type	Destination	Direction Probabilities				Herd Probabilities	
		$p_{intended}$	p_{left}	p_{right}	$p_{opposite}$	p_{join}	p_{leave}
Type 1	Home	0.6	0.2	0.2	0.0	0.7	0.1
	Forage	0.52	0.16	0.16	0.16		
	Random	0.25	0.25	0.25	0.25		
Type 2	Home	0.9	0.05	0.05	0.0	0.7	0.3
	Forage	0.8	0.1	0.1	0.0		
	Random	0.25	0.25	0.25	0.25		
Type 3	Home	0.6	0.2	0.2	0.0	0.5	0.5
	Forage	0.8	0.1	0.1	0.0		
	Random	0.25	0.25	0.25	0.25		

$1 : \langle 0.6, 0.4 \rangle$ for $\langle T_1, T_2 \rangle$ $2 : \langle 0.6, 0.4 \rangle$ for $\langle T_1, T_2 \rangle$
$3 : \langle 0.6, 0.4 \rangle$ for $\langle T_1, T_3 \rangle$ $4 : \langle 0.6, 0.4 \rangle$ for $\langle T_2, T_3 \rangle$
$5 : \langle 0.6, 0.4 \rangle$ for $\langle T_2, T_1 \rangle$ $6 : \langle 0.6, 0.4 \rangle$ for $\langle T_3, T_2 \rangle$

Observer calculates the expected rewards for the Stochastic belief update model using equation 7, as follows:

$$ER_1 = 0.6 \times 1 + 0.4 \times 2 = 1.4$$
$$ER_3 = 0.6 \times 1 + 0.4 \times 3 = 1.8$$

Similarly $ER_2 = 1.4$, $ER_4 = 2.4$, $ER_5 = 1.6$ and $ER_6 = 2.6$. Using these Expected Rewards, the weights for each target are calculated using equation 7 as follows:

$$\omega_1 = \frac{1.4}{1.4+1.4+1.8+2.4+1.6+2.6} = \frac{1.4}{11.2} = 0.125.$$

Similarly $\omega_2 = 0.125$, $\omega_3 = 0.16$, $\omega_4 = 0.214$, $\omega_5 = 0.143$ and $\omega_6 = 0.233$. From the calculated ω_i's, using the target positions Po_j, the weighted mean WM_i is computed as follows and plugged in equation 8 to compute the observer's destination.

$$WM_i = \sum_{j=1}^{6} \omega_j \times Po_j, \tag{9}$$

Although we incorporated multiple target types and uncertainty into mean point algorithm, computing optimal altitude level for the observer is not feasible since BRLP-CTO calculates the observer's destination through update equation 1 which solely depends on mean position of targets. Observer related attributes like altitude and remaining battery power cannot be modeled as mean positions (or other target related information). In general, including such observer related attributes into decision making would require satisfying multiple objectives and constraints. Hence the algorithm can only decide on the new destination, while the observer stays at the same altitude level. For the case when the observer is not observing any targets, the observer gets to choose a random position and a random altitude level.

7. Experiments

7.1. Target Modeling

We build two different target models namely Random Model and Markov Chain Model for target behavior. In the Random Model, all animals move randomly from one point to another. Animals get to pick their next destination and once they reach the destination, they compute a new point to reach. Hidden Markov Models are commonly used to model animal behaviors [22, 23] and can be complex in nature depending on the purpose. We present here details of a simpler Markov Chain model used in our experiments. The model consists of 3 states, **Go Home**, **Go Forage** and **Go Random**. Home and Food locations are randomly generated for each animal type at the start of the simulation. Animals choose to move in the direction of home/food, if the state is Go Home/Go Forage. When the state is Go Random, animals move in a random direction. To handle navigation uncertainty in animals, we use another set of Markov Chains to compute the actual direction in which animals would move based on their current state and the intended direction. Animals can move individually or form a herd. For simulation purposes, a herd is treated as a single entity and the animals of a herd act in unison. A lone animal can join a herd with a probability if the herd type is same as its type and the distance between the herd and itself is less than a predefined constant ϵ. Similarly, an animal in a herd can leave the herd anytime with certain probability. Maximum herd size is limited to 4 animals.

The parameters we used in Markov Chain model are summarized in Table 1. An animal or a herd identifies its destination using the two Markov Chains. While the first Markov Chain decides the next state animal should reach to, the second Markov Chain then decides whether the animal or herd should proceed in the intended or some other direction. In the first Markov

Table 2. Observer scores for Markov Chain based target model

Observer Strategy	Actual Score	Expected Score	True Positive Score (TP)	True Negative Score (TN)	Ratio (TP/TN)
SM + BRLP-CTO	67570.96	92250.92	37829.07	54421.85	0.69
MCDA-CTO RABC	41056.06	58237.71	27955.96	30281.74	0.92
MCDA-CTO BABC	55516.37	78837.13	39813.6	39023.52	1.02
MCDA-CTO QR	40615.93	52371.8	29230.53	23141.26	1.26
MCDA-CTO BAA	60372.26	90613.56	45879.98	44733.58	1.02

Chain, all states are equi-probable from a given state, i.e. the next state (Go Home, Go Forage or Go Random) for the animal or herd is picked with the probability $\langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$. Once next state is decided, the required direction is identified and labeled as intended. The $p_{intended}$, p_{left}, p_{right}, $p_{opposite}$ values for each target type are as summarized in Table 1 where p_{left} and p_{right} indicate left and right directions w.r.t. the intended direction, p_{join} captures the probability with which an animal joins a herd and p_{leave} captures the probability with which an animal leaves the herd. Animal herds, their home and food locations are randomly generated.

7.2. Experimental setup

For purposes of experiments, we assume that the observers and targets are operating in a rectangular field with a width and height of 150 $\times$ 150 units. Observers have 4 different levels of altitudes, with sensor range at lowest level being 10 units and 20 units at the highest. The total number of observers are 10. We assume 3 different types of targets with total number of targets to be 25, with a distribution $\langle 12, 8, 5 \rangle$ across the three types. The reward values for the target types are $\langle 1, 5, 10 \rangle$ respectively i.e. the observer receives 1 unit of reward for observing a target of type 1 while it receives 5 and 10 units for observing targets of type 2 and 3 respectively. Maximum time for targets to reach their destination is 100 time-steps. If a target reaches its destination within the time limit, it computes a new destination and moves towards it. Observers compute their destination once in every 10 time-steps even if they have reached their destination beforehand, to ensure that all observers make decisions at same time (needed when communication between the observers is possible). The speed of observers and targets is assumed to be 1 unit per time-step. We set the time required for the observer to shift from an altitude level to its next level as 2 time-steps, during which the observer cannot make any observation. Hence, the maximum time the observer needs to shift altitude levels is 6 time-steps.

Each experiment consists of 1500 time-steps and is simulated 30 times. Positions of observers and targets are randomly generated at the start of simulation. For each simulation run we calculated all the four scores and the mean of each score across the 30 runs is presented. A target observed by multiple observers is counted only once. All the experiments were performed on MASON simulation toolkit [24], exploiting its internal threading mechanism to run the observers and targets in parallel.

7.3. Results

We present simulation results in Table 2, where the target uses the Markov Chain model. For brevity purposes, we omit results for Random Target model, but the observers performance remains similar for both the models. The table compares the performance of the four action picking strategies for MCDA-CTO with the BRLP-CTO algorithm (note that SM + BRLP-CTO refers to stochastic belief update model). We present five scores in total: four scores for each observer strategy namely Actual, Expected, True Positive (TP) and True Negative (TN) score and the **ratio of True Positive to True Negative (i.e. TP/TN) as the fifth score**. A higher (TP/TN) ratio implies a higher level of awareness i.e. the observer knows better the true type of the target it is observing. We focus here on the Actual Score and (TP/TN) since Actual Score reflects the ground truth while the (TP/TN) ratio reflects the belief observer has about ground truth. The table shows that Actual Score is significantly higher for BRLP-CTO since a BRLP-CTO observer typically ends up staying for a long time in higher or highest altitude levels where the sensor range is higher and has the best chance to observe more targets. However the observer has a much higher uncertainty in the observations which do not get reflected in the Actual score. We also tested the case where the observer always stays in the lowest altitude for both BRLP-CTO and MCDA-CTO and found that the MCDA-CTO (BABC and BAA in particular) has a higher Actual score (results omitted for space reasons).

We also presented TP scores in Table 2 and found that deterministic versions of MCDA-CTO are better than BRLP-CTO implying that the MCDA-CTO observer observed the targets with a better knowledge

Table 3. Penalties vs Performance

Penalty	MCDA-CTO RABC		MCDA-CTO BABC		MCDA-CTO QR		MCDA-CTO BAA	
	Actual Score	Ratio	Actual Score	Ratio	Actual Score	Ratio	Actual Score	Ratio
None	46137.66	0.97	781037.7	1.02	49911.20	1.05	80648.26	1.05
P_1	45147.83	0.95	89368.20	0.91	58152.50	0.83	90651.06	0.95
P_2	45205.73	0.99	43680.16	2.40	43120.96	1.80	43247.80	2.24
$P_1 + P_2$	46059.90	0.95	62161.93	1.03	45490.30	1.25	64797.86	1.01
$P_2/2$	45767.73	0.96	43316.13	2.35	44463.13	1.38	43083.53	2.20
$P_1 + (P_2/2)$	46525.16	1.00	91674.30	0.93	49289.83	0.99	90172.23	1.00

Table 4. MSE of different adversaries

Adversary Model	SM + BRLP-CTO	MCDA-CTO RABC	MCDA-CTO BABC	MCDA-CTO QR	MCDA-CTO BAA
Linear Regressor	7.73	12.64	14.55	10.01	17.32
Elastic Net	7.92	13.21	15.54	10.62	18.21
Logistic Regression	16.87	27.79	25.10	22.67	33.51
Neural Net Regressor	6.58	13.51	10.16	10.26	11.49

of their true types. This is also reflected in the (TP/TN) ratios, where the ratio is significantly higher for MCDA-CTO (all the versions) than for BRLP-CTO. This shows that **MCDA-CTO takes actions with a lot more certain information**. As monitoring applications become automated, observers need to have a better knowledge of true target type since their course of action depends on it (unlike when human is in loop).

To summarize, there is a trade-off between the Actual Score and the (TP/TN) ratio i.e., improving one seems to deteriorate the other. By changing the penalty value or levying some instead of all, MCDA-CTO algorithm can be tailored to achieve desired results (Note that BRLP-CTO solution is invariant to these penalties and hence is not affected). We summarize the penalty terms used and the performance of MCDA-CTO algorithm in Table 3. In the table, None corresponds to no penalty i.e. observer takes decisions solely based on expected reward calculations. P_1 implies only penalty P_1 is levied, that discourages the observer from changing altitude. Hence the observer stays longer in higher altitudes resulting in better score than BRLP-CTO, but has a poor ratio. Similarly, P_2 implies only penalty P_2 is levied, that forces the observer to stay in lower altitude and hence has better ratio but poor score. (P_2/2) implies that only half the penalty of P_2 i.e. $aH \times (-1.5)$ is levied. Likewise, P_1 + (P_2/2) is another penalty setting while P_1 + P_2 is the presented original setting (different simulation run).

7.4. Adversary Modeling

To show that MCDA-CTO indeed does better randomization than BRLP-CTO, we modeled different adversaries (using different regression methods) that try to predict the destination of the observer based on the observer's position and the targets within its sensor range. We assume that the adversaries have global knowledge of all the targets and observers at any point of time. Each adversary is trained on more than 6000 instances of decision making by the observers and tested on 1000 new instances. We calculate **Mean Squared Error (MSE)** which is the mean of sum of squared distance between the destination predicted by the adversary and the actual destination chosen by the observer. Table 4 generated using the Scikit library [25], summarizes the results. The table shows that deterministic procedures MCDA BAA and MCDA BABC perform better than MCDA RABC and MCDA QR which randomize over the randomness generated by the clustering algorithm. This maybe because, the best action is chosen from the set of n actions where as the random action is chosen from the best cluster of say q actions ($q < n$). Note that all variants of MCDA-CTO have a higher MSE than BRLP-CTO, implying that the adversary is unable to predict the observer's destination as well as when the observer uses BRLP-CTO, showing that a better randomization was achieved.

8. Conclusions and Future Work

We present here a clustering based algorithm **for wildlife monitoring** named MCDA-CTO, to handle multiple criteria that arise due to reasoning over the different clusters of targets and also due to environmental factors such as altitude, battery power etc. We then modified the MCDA-CTO algorithm for the scenario where UAVs are observers to perform

a high quality observation of targets in the presence of observational uncertainty along with a better randomization. We then provided comparison results against state of the art algorithm BRLP-CTO. In order to generate these results we first had to extend the BRLP-CTO algorithm to enable handling of multiple target types and observation uncertainty resulting in the Stochastic belief update model. The extensions still do not allow for combinatorial reasoning over different clusters that MCDA-CTO performs. Our experiments show that MCDA-CTO provides significant benefits in terms of rewards and randomization.

Future Work: In the future, we plan to explore Reinforcement Learning based techniques for the wildlife monitoring problem with the aim arrive at an optimal solution with minimal inputs from the DMs.

References

[1] W. W. F. for Nature, "A warning sign from our planet: Nature needs our support," 2018. https://www.wwf.org.uk/updates/living-planet-report-2018.

[2] WildlifeACT, "Wildlife tracking and monitoring," 2012. https://wildlifeact.com/about-wildlife-act/wildlife-tracking-and-monitoring.

[3] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, "Are unmanned aircraft systems (uass) the future of wildlife monitoring? a review of accomplishments and challenges," *Mammal Review*, vol. 45, no. 4, pp. 239–252, 2015.

[4] E. Bondi, F. Fang, M. Hamilton, D. Kar, D. Dmello, J. Choi, R. Hannaford, A. Iyer, L. Joppa, M. Tambe, *et al.*, "Spot poachers in action: Augmenting conservation drones with automatic detection in near real time," 2018.

[5] O. M. Cliff, R. Fitch, S. Sukkarieh, D. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system.," in *Robotics: Science and Systems*, 2015.

[6] T. França da Silva, J. L. Alves Leite, R. J. Campos Ferro Junior, L. Ferreira da Costa, R. Pinheiro de Souza, J. P. Bernardino Andrade, and G. A. Lima de Campos, "Smart targets to avoid observation in cto problem," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1958–1960, International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[7] J. Hu, L. Xie, K.-Y. Lum, and J. Xu, "Multiagent information fusion and cooperative control in target search," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1223–1235, 2013.

[8] S. Jacobi, C. Madrigal-Mora, E. León-Soto, and K. Fischer, "Agentsteel: An agent-based online system for the planning and observation of steel production," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 114–119, ACM, 2005.

[9] C. V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 137–144, ACM, 2003.

[10] A. Khan, B. Rinner, and A. Cavallaro, "Multiscale observation of multiple moving targets using micro aerial vehicles," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 4642–4649, IEEE, 2015.

[11] R. Rysdyk, "Unmanned aerial vehicle path following for target observation in wind," *Journal of guidance, control, and dynamics*, vol. 29, no. 5, pp. 1092–1100, 2006.

[12] B. B. Werger and M. J. Matarić, "Broadcast of local eligibility for multi-target observation," in *Distributed autonomous robotic systems 4*, pp. 347–356, Springer, 2000.

[13] L. E. Parker, "Cooperative robotics for multi-target observation," *Intelligent Automation & Soft Computing*, vol. 5, no. 1, pp. 5–19, 1999.

[14] A. Kolling and S. Carpin, "Cooperative observation of multiple moving targets: an algorithm and its formalization," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 935–953, 2007.

[15] B. Jung and G. S. Sukhatme, "Cooperative multi-robot target tracking," in *Distributed Autonomous Robotic Systems 7*, pp. 81–90, Springer, 2006.

[16] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous robots*, vol. 12, no. 3, pp. 231–255, 2002.

[17] S. Luke, K. Sullivan, L. Panait, and G. Balan, "Tunably decentralized algorithms for cooperative target observation," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 911–917, ACM, 2005.

[18] R. Aswani, S. K. Munnangi, and P. Paruchuri, "Improving surveillance using cooperative target observation," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[19] Y. De Smet and L. M. Guzmán, "Towards multicriteria clustering: An extension of the k-means algorithm," *European Journal of Operational Research*, vol. 158, no. 2, pp. 390–398, 2004.

[20] J. R. Figueira, V. Mousseau, and B. Roy, "Electre methods," in *Multiple Criteria Decision Analysis*, pp. 155–185, Springer, 2016.

[21] J.-P. Brans and B. Mareschal, "Promethee methods," in *Multiple criteria decision analysis: state of the art surveys*, pp. 163–186, Springer, 2005.

[22] I. L. Macdonald and D. Raubenheimer, "Hidden markov models and animal behaviour," *Biometrical Journal*, vol. 37, no. 6, pp. 701–712, 1995.

[23] R. Langrock, R. King, J. Matthiopoulos, L. Thomas, D. Fortin, and J. M. Morales, "Flexible and practical modeling of animal telemetry data: hidden markov models and extensions," *Ecology*, vol. 93, no. 11, pp. 2336–2342, 2012.

[24] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan, "Mason: A new multi-agent simulation toolkit," in *Proceedings of the 2004 swarmfest workshop*, vol. 8, p. 44, 2004.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.