Cooperative Swarm Optimization: a teamwork-based approach to reduction of uncertainty

Kyle David DeMedeiros* and Thomas A. Wettergren†

Naval Undersea Warfare Center

1176 Howell Street

Newport, RI 02841

Email: kyle.demedeiros@navy.mil*, thomas.wettergren@navy.mil†

Abstract—We describe a new metaheuristic, called cooperative swarm optimization (CSO), which manages the efficient use of a group of unmanned agents performing cooperative operations. In this work we are concerned with the problem of group exploration of an uncertain environment. The CSO algorithm was developed to perform this exploration with limited inter-agent coordination and no central oversight. CSO is a swarm search algorithm that uses the mechanics of velocity to move each agent about the information space. In this way, CSO is similar to particle swarm optimization (PSO) however, CSO differs from PSO in how information is used from other agents and how the information is combined. In particular, the individual agent velocity is affected by the current agent information, the personal history of the agent, and the group's information and is weighted based on the entropy rate of change of the information space, allowing each agent to take into account the dynamic nature of the information space. New information is weighted highest, followed by changes to currently known information. Information that does not change is rated lowest. We illustrate the derivation of the algorithm and show numerical examples to demonstrate its effectiveness for use in cooperative unmanned underwater vehicle missions.

Keywords: particle swarm optimization, autonomous agents, swarms, metaheuristics, cooperative behaviors

I. INTRODUCTION

In this work we are concerned with the problem of group exploration of an uncertain environment. This is an important problem in the undersea domain, where unmanned systems are increasingly being used to manage potentially dangerous tasks and where group autonomy is of great interest to help facilitate large-scale tasking. There is a lot of focus on group autonomy using nature-inspired techniques, especially in the undersea domain [1],[2],[3],[4],[5]. These techniques are attractive because they have the benefit of being evaluated by the organisms using them in environments of interest. This allows unmanned systems to leverage generations of naturally tested techniques. Nature-inspired techniques also show redundancy of use (for instance, swarming on land, in the air, and under the sea), which further proves their usefulness.

Since the inception of modern computing, people have had the desire to emulate nature in mathematics. This emulation can take many forms; from mimicry of animal and insect behaviors [6], to human-like interactions and decision processes [7]. Evolutionary algorithms (EA) try to mimic the natural space by mimicking the processes that aid in the evolution in living organisms [8]. They do this by attempting to mimic the

processes that cause offspring to have traits similar to their parents [8].

Our work focuses on the implementation of swarm techniques that are characterized by the need for limited inter-agent communication and no central oversight. To this end, we have developed a swarm-based metaheuristic called Cooperative Swarm Optimization (CSO). CSO is a swarm search algorithm that uses group and individual information to adjust the individual agent velocity and move each agent about the information space. In this way, CSO is similar to particle swarm optimization (PSO) [9]; however, CSO differs from PSO in how information is used from other agents and how the information is combined.

CSO utilizes Multi-Agent Autonomy (MAA) to achieve the goal with a loosely coordinated group of agents. Multi-agent autonomy (MAA) involves a group working together through a set of common autonomous rules/behaviors to achieve some goal. One of the practical limitations in group autonomy is communication of information and the requirement to have system robustness against failure. CSO leverages swarm mechanics to provide the coordination of the group, and does so in a manner that reduces the amount of information needed to cooperate intelligently.

The numerical examples in this paper are based on searching for a target of unknown disposition in an uncertain environment. The information space is a spatial map which corresponds to the likelihood a target is at a given location on the map. The agents read in the data (sensor readings from the local environment) and use that data to calculate the rate of change of the uncertainty in the data values of the search area. The information uncertainty change rates from other agents, along with the agents' locations, are sent to every other agent. This allows each agent to create an internal map of the system and the information feeds into the CSO algorithm to pull the agent to a new location.

The remainder of this paper is organized as follows. Section II describes the concepts of swarm mechanics and the PSO algorithm. Section III develops the CSO algorithm and explains how it works. Section IV outlines the computational experiments performed and the metrics used to evaluate CSO against PSO. Section V describes the results seen in the experimentation. The paper concludes with a discussion of lessons learned and future work.

II. SWARM MECHANICS

When one thinks of a swarm, one of the most common images conjured is that of a swarm of insects (bees, locusts, etc.). This concept can be mirrored in groups of autonomous agents by looking not at what a swarm looks like, but at how it behaves. A swarm is a group of simple objects interacting with the environment and each other in order to create complex behaviors [10]. When looking at swarms in this context, their capabilities as tools for problem solving become apparent. Swarms have some benefits that other metaheuristics (those that rely on specific groupings, patterns, and formations of agents) do not:

- Any agent can perform any task.
- Information is shared throughout the group.
- Group survivability (if one agent fails, the group can continue the mission).

These benefits ensure that with swarms, there is generally no single point of failure. Furthermore, swarms tend to allow for very simple individual agent behaviors, which create robust solutions in highly dynamic environments.

The CSO algorithm has its roots in another swarm-based metaheuristic called particle swarm optimization (PSO) [9]. PSO was created by Kennedy and Eberhart in the mid 1990's after observing a simulation of a simplified social model of the movement patterns of a flock of birds. This social model was then generalized to create a general purpose metaheuristic optimization technique. PSO searches the parameter space by using a set of particles, which act as agents in a multi-agent autonomy setting. For the purposes of explanation, the particles will be treated as moving points in 2-dimensional space, but the usage of PSO is not restricted to 2-dimensions. Particles move through space, continually being evaluated based on a pair of meaningful solutions. As the particles move around, their current value (based on an objective function) is modified based on their current position in the parameter space vs a global best solution (the best solution seen out of every particle) and a local solution (the best solution the individual particle has visited as it was moving around the parameter space). Particles are attracted to both their local best solution and the global best solution, and weights determine how attractive each solution is to the particle. In this way, mating does not occur; the updating of the particles acts as a mutation which drives the move toward a meaningful (though not necessarily optimal) solution.

Assume the system operates in a workspace $\mathcal{W}$ that is a closed subset of the plane ($\mathcal{W} \subset \mathbb{R}^2$). The system consists of K agents labeled as $k = \{1, \dots, K\}$. The system operates over a finite time interval of length T_f that has been divided into equally sized step intervals t_i of length δt , such that $t_i = t_0 + i\delta t$. Each particle k has a current location x_i^k , a personal best location (known as p_{best}^k), and knows of the group's best location (known as g_{best}). The particle also has a velocity v_i^k as well as a momentum w that weights the importance of maintaining the current velocity v_i^k . Each particle also has a confidence for its p_{best}^k (known as c_1) and g_{best}

(known as c_2). These confidences, along with randomness, affect the movement of the particles. We assume each particle uses the same values of momentum and confidence, such that these parameters have no k -dependence. Randomness is input into the system to allow for sufficient movement about the parameter space to allow for better g_{best} values to occur.

These considerations are put together to create the following agent velocity update rule [11] that forms the core of PSO:

$$v_i^k = w v_{i-1}^k + u_1 c_1 (p_{best}^k - x_{i-1}^k)/\delta t + u_2 c_2 (g_{best} - x_{i-1}^k)/\delta t \quad (1)$$

where x_{i-1}^k is the position of the k -th agent at time step $(i-1)$ (current time) and v_{i-1}^k is its velocity. The parameters u_1 and u_2 are random parameters, chosen as samples from a uniformly random variable over the range $[0, 1]$, and δt is the time interval between time step $(i-1)$ and time step i . The new position x_i^k of the agent is shown in equation (2) as

$$x_i^k = x_{i-1}^k + v_i^k \delta t \quad (2)$$

Thus, each agent adjusts its new position according to some combination of inertia, its own previous best location, and the best location seen previously by any member of the group.

A variety of values of w , c_1 , and c_2 were examined in [9], but it was determined that $w = 1$ and $c_1 = c_2 = 2$ were sufficient for the initial examples. In [11], the algorithm from [9] was compared to the genetic algorithm. It was determined that the values of $w = 0.5$ and $c_1 = c_2 = 1.5$ were better values for more general purposes and are generically used in PSO implementations.

III. COOPERATIVE SWARM OPTIMIZATION

Cooperative Swarm Optimization (CSO) is a new swarm-based metaheuristic inspired by Particle Swarm Optimization (PSO) [9]. CSO utilizes information obtained from each agent along with information shared between the agents to reduce information space uncertainty. This cooperative reduction of information space uncertainty allows CSO to create near-optimal solutions to optimization problems. In a spatial context, these problems typically involve finding hidden objects or tracking environmental parameters. Like PSO, CSO uses both shared information from the other agents along with the individual agent information to update the agent's velocity. This velocity is weighted by three factors:

- The agent's inertia
- The agent's personal history
- The group's reported information about the information space

To accomplish this cooperative goal, each agent traverses the information space by considering the agent's current information (what it can see right now), the agent's personal history (where it's been and the relevance of the information at that point), and the group's information (what each group member sees and where they are located within the information space). This information is utilized to "pull" the agent to its next objective. Unlike PSO, CSO's weighting is neither a fixed

number, nor a empirically-determined number, but rather a measure of change of the data in the information space, which is determined directly from the information space and based on the reports of agents who have searched the information space previously.

To curb the potential of a large influx of data being sent to and from each agent, the agent location and the local value of information uncertainty are the only parameters being sent to an agent from other agents in the group. In this way, the CSO algorithm utilizes limited bandwidth to facilitate maximum information sharing.

A. Modeling of Measurement Uncertainty

The CSO algorithm uses a concept of reduction of uncertainty to be the primary goal of the agents, whether they be physical entities (such as UAV's) or informational entities (such as in a numerical optimization setting). For our unmanned vehicle context, the workspace will consist of a spatial domain $\mathcal{W}$ that is a closed subset of the plane ($\mathcal{W} \subset \mathbb{R}^2$). The system consists of K agents labeled as $k = \{1, \dots, K\}$. The system operates over a finite time interval of length T_f that has been divided into equally sized step intervals t_i of length δt , such that $t_i = t_0 + i\delta t$.

At the i -th time step, agent k obtains a measurement $\alpha_i^k(x)$ corresponding to a reading that it makes at its position $x \in \mathcal{W}$. The agent uses that measurement to make an update to its internal understanding of the measurement space, specifically by updating $\eta_{i-1}^k(x)$, which is the *expected measurement value* that the k -th agent expects to see at location x based upon information obtained up to time step t_{i-1} . We have an additional spatial variable $\lambda_i(x)$ which represents the *uncertainty in the information* that has been obtained at location x . This uncertainty is reflective of both the amount of effort that has been applied to location x (i.e. how many agents have previously made measurements there) as well as the amount of discrepancy against their expectations that the various agents have found when making measurements at that location. We require that the value of the location uncertainty fall in a bounded range $0 < \lambda_i(x) \leq 1$ where a value of $\lambda_i(x) = 1$ corresponds to complete uncertainty in the information at the location (an entropic condition) and the information tends to a specific fixed certain value as $\lambda_i(x) \rightarrow 0$. In this was the expected measurement values $\eta_i^k(x)$ can represent things such as simple physical parameters (i.e. temperature, salinity, etc.) or they can represent target likelihoods in search and tracking applications.

While the agents do not share their direct measurements nor their individual understanding of the space, they do share any updates that they make to the local uncertainty of each location that they visit to make a measurement. Thus the uncertainties $\lambda_i(x)$ do not have an agent-dependence. Also note that they are the only information shared between agents, and can be shared on a coarser time and/or spatial scale than that of the individual agent maps $\eta_i^k(x)$.

To initialize the system process we let all of the expected measurement values for all of the agents be set to a known

(known to all the agents) nominal level $\eta_0(x) = \eta_i^k(x)$ at $i = 0$ for all k . This can be the assumed value of a physical quantity to be measured or the *a priori* target location likelihood values for a search problem. The only requirement on the values of $\eta_0(x)$ is that $\eta_0(x) > 0$. Furthermore, we initialize the location uncertainties to begin at the maximum uncertainty, such that $\lambda_0(x) = 1$ for all x .

Given these initializations, we can compute the changes to the expected measurement value of the individual agents according to the following update rule:

$$\eta_i^k(x) = \lambda_{i-1}(x) \cdot \alpha_i^k(x) + (1 - \lambda_{i-1}(x)) \cdot \eta_{i-1}^k(x). \quad (3)$$

Note that this updating rule is a simple weighting of the measured value and the agent's prior expectation at that location. The weighting values have been chosen such that the weighting biases strongly towards the new measurements when there is high uncertainty (low confidence) in the prior expectations (as $\lambda_{i-1}(x) \rightarrow 1$), and conversely the weighting biases strongly towards the prior expectations when there is low uncertainty (high confidence) in the prior expectations (as $\lambda_{i-1}(x) \rightarrow 0$).

In addition to updating the expected measurement value for the individual agent, a measurement will also cause the agent to update the value of the shared local uncertainty $\lambda_i(x)$ at the measurement location x . To perform this update it is convenient for the agent to first compute the *observed measurement discrepancy* $\psi_i^k(x)$, which is given by

$$\psi_i^k(x) = \left| \frac{\alpha_i^k(x) - \eta_{i-1}^k(x)}{\eta_0(x)} \right| \quad (4)$$

This observed measurement discrepancy is the difference in the measurement and the sensor's expected measurement, normalized by the initial value of the expected measurement. Clearly a large value of $\psi_i^k(x)$ corresponds to a measurement that is vastly different than expectations, whereas a small value corresponds to a measurement that meets the current expectations. We use this discrepancy value to update the shared local uncertainty through the following update rule:

$$\lambda_i(x) = \lambda_{i-1}(x) [1 - \exp(-\psi_i^k(x))] \quad (5)$$

Thus, when there are multiple measurements at a given location that meet the expectations, the value of the local uncertainty decreases rapidly. When there are measurements that are further from the expectations, the decrease in uncertainty is slower. Note that we are always decreasing uncertainty as more measurements are being made. This is because there is always a benefit to new information being obtained through taking measurements at a location.

If there is reason to believe that the underlying measurement quantities may be dynamic, then a simple *fading memory* can be applied to the shared values of $\lambda_i(x)$ via

$$\lambda_i(x) \mapsto \gamma + (1 - \gamma)\lambda_i(x) \quad (6)$$

for some value of $0 \leq \gamma < 1$ (where $\gamma = 0$ corresponds to no fading). The fading memory has the effect of increasing

all of the uncertainty values in a proportional manner. This fading memory can be applied intermittently or periodically depending on the dynamics that it is desired to account for.

B. The CSO Equation

CSO moves individual agents by direct adjustment of the agent velocity at each time step. We note that there is a potential drawback to the use of the information uncertainty parameter $\lambda_i(x)$ to drive the agent velocities. In particular, there is a tendency to “spread out” all agents regardless of the underlying importance of specific regions of the space. This phenomenon is alleviated by applying a bias to the initial expectations $\eta_0(x)$ that artificially creates a deviation from the expected measurement in locations of the space that have some other inherent importance for being visited by the CSO agents.

Using the updated values of $\lambda_i(x)$, λ_{best}^k and x_{best}^k , the CSO algorithm provides motion plans for each agent that allows the group to efficiently reduce the overall information uncertainty in the workspace. The specific form of velocity update that is used by CSO is given as follows:

$$v_i^k = w v_{i-1}^k + u_2 \lambda_{best}^k (x_{best}^k - x_{i-1}^k) / \delta t + u_3 \left[\frac{1}{K} \sum_{j=1}^K \lambda_i(x_{i-1}^j) (x_{i-1}^j - x_{i-1}^k) / \delta t \right] \quad (7)$$

where x_{best}^k is the best location obtained in the prior history by agent k in terms of the maximal value of $\psi(x)$ that was achieved. The value λ_{best}^k is the value that was achieved at that time step. Specifically, these are given by

$$x_{best}^k = \left\{ x_{i^*}^k : i^* = \underset{i}{\operatorname{argmax}} \psi_i^k(x) \right\} \quad (8)$$

and

$$\lambda_{best}^k = \lambda_{i^*}(x_{best}^k) \quad (9)$$

We note that the agent velocity given by equation (7) has a first term that is comprised of a constant weighting (w) that both multiplies the previous agent velocity. Thus, we refer to this first term as an *inertial term*. The second term of equation (7) has a random weight (u_2) but has additional weighting based on the prior best of the current agent ($\delta \lambda_{best}^k$) and multiplies the difference between that location and the current agent location. This term provides a pull towards the best that the individual agent has ever performed (in terms of getting most gain in a step). The third term of equation (7) also has a random weight (u_3) but it consists of a weighted sum of the difference in location between the current agent and all other agents, with the weightings in the sum being given by the current performance of each agent. Thus the third term provides a pull towards a weighted center-of-mass of the entire group of agents. As in PSO, the CSO algorithm performs this balance between three competing “pulls” in the velocity space. To actually compute the new location of the agent at the next time step, it is simple to apply the following:

$$x_i^k = x_{i-1}^k + v_i^k \delta t \quad (10)$$

Equations (7) and (10) create the CSO algorithm that is exercised on each of the agents in the group. Note that the only information sharing required between agents is their position x_{i-1}^j and corresponding performance $\lambda_i(x_{i-1}^j)$. It is not necessary that these be shared at a fine resolution time step, as an individual agent can change velocity according to the frequency that it receives updates.

C. The CSO Algorithm

The CSO equation as given in equation (7) is the main part of the CSO algorithm. The algorithm implementation of the CSO equation produces an iterative process which utilizes MAA to create individual agent behaviors that allow the group of agents to collectively be effective in traversing the information space. The algorithm allows each agent’s information uncertainty map $\lambda_i(x)$ to be updated with information from throughout the group. This update, in turn, will update the individual agent’s expected measurement value map $\eta_i^k(x)$ at the next time step. The CSO algorithm that is implemented for each agent is given in Algorithm 1.

Algorithm 1 CSO Algorithm

```

Input a new value of  $\alpha_i^k(x)$ ;
Compute  $\psi_i^k(x)$  from  $\alpha_i^k(x)$  and  $\eta_{i-1}^k(x)$ ;
Compute  $\eta_i^k(x)$  from  $\alpha_i^k(x)$ ,  $\eta_{i-1}^k(x)$  and  $\lambda_{i-1}(x)$ ;
Compute  $\lambda_i(x)$  from  $\lambda_{i-1}(x)$  and  $\psi_i^k(x)$ ;
for  $k = 1$  to  $K$  do
  if  $\psi_i(x_{i-1}^k) \geq \psi_{best}^k$  then
     $\psi_{best}^k = \psi_i(x_{i-1}^k)$ 
     $x_{best}^k = x_{i-1}^k$ ;
     $\lambda_{best}^k = \lambda_i(x_{i-1}^k)$ ;
  end if
end for
Update  $\lambda$  map and share  $\lambda_i(x)$  with other agents;
Compute  $v_i^k$  from the CSO equation;
if  $v_i^k > v_{max}$  then
   $v_i^k = v_{max}$ ; {for restricting agent speed}
end if
Set  $x_i^k = x_{i-1}^k + v_i^k \delta t$ 

```

IV. EXPERIMENTS

Here we describe the numerical experiments used to compare the CSO and PSO algorithms. A single area of uncertainty is being used to compare the two algorithms. This is represented by a single “bump” in the information space as seen in Fig. 1.

In addition, the size of the swarm can have an effect on its success. Two different agent population sizes were chosen:

- 5 agents (representative of a small team of unmanned vehicles).
- 15 agents (a representation of a large group of unmanned vehicles).

The sizes for the agent population were chosen to represent different potential mission types. For instance, a 5-agent group

Fig. 1. A representation of the information space.

might be used for a communications relay and a 15-agent group might be used for a detailed sweep of an area.

As mentioned in Section II, The w , $c1$, and $c2$ weights used in PSO are fixed. This work will use the values of 0.5, 1.5, and 1.5 respectively for the PSO weights in keeping with the findings from [11]. For CSO, testing has shown a value of 1 to be effective for w . Each set of agents will be run 10 times and each CSO and PSO test within a run have the same start positions to avoid bias based on start positions of the agents. New start positions are randomly generated for each run. Start velocities are randomly generated for each test.

The goal of this comparison is to see a decrease in uncertainty. This will be seen in a results table comparing the uncertainty of CSO vs PSO for each run. The total uncertainty within the map at the end of each experiment should be lower than at the start of each experiment. In this way, a comparison between the two algorithms can be made.

V. RESULTS

Here we discuss the results of the numerical experiments. As seen in Table I, CSO had a markable improvement in reduction of uncertainty when compared to PSO. On average, CSO reduces uncertainty by approximately 9% more than PSO with 5 agents, and approximately 8% with 15 agents. Visually, this can be seen in Figs.2 through 5. With CSO, there is visibly more spread among the agents, leading to increased reduction of uncertainty. With PSO, the agents tend to stay clumped together more, reducing their effectiveness.

Fig. 2. 5 agents running CSO.

Fig. 3. 5 agents running PSO.

Fig. 4. 15 agents running CSO.

TABLE I
RESULTS OF 10 SIMULATION RUNS OF CSO AND PSO

Run	5 Agents		15 Agents	
	CSO	PSO	CSO	PSO
1	0.849	0.918	0.666	0.774
2	0.850	0.967	0.703	0.756
3	0.845	0.950	0.724	0.783
4	0.858	0.952	0.688	0.804
5	0.859	0.955	0.710	0.773
6	0.876	0.915	0.712	0.794
7	0.859	0.956	0.699	0.797
8	0.851	0.950	0.715	0.802
9	0.844	0.936	0.746	0.789
10	0.868	0.931	0.690	0.810
avg	0.856	0.943	0.705	0.788

Fig. 5. 15 agents running PSO.

VI. CONCLUSION

We have developed a swarm-based optimization algorithm called Cooperative Swarm optimization (CSO) which has been used for reduction of uncertainty in an information space. We have shown a marked improvement in reduction of uncertainty when compared to Particle Swarm Optimization (PSO); a well known swarm-based optimization technique. Future work will include multiple areas of uncertainty and moving areas of uncertainty.

ACKNOWLEDGMENT

This work has been supported by the Office of Naval Research code 333.

REFERENCES

- [1] C. T. Cheng, K. Fallahi, H. Leung, and C. K. Tse, "A genetic algorithm-inspired uuv path planner based on dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 1128–1134, Nov 2012.
- [2] B. Prasad, A. Agrawal, V. Viswanathan, A. R. Chowdhury, R. Kumar, and S. K. Panda, "Genetic algorithm based intelligent station-keeping for spherical underwater robot," in *IEEE Underwater Technology (UT)*, pp. 1–6, Feb 2015.

- [3] M. Martz and W. L. Neu, "Multi-objective optimization of an autonomous underwater vehicle," in *OCEANS 2008*, pp. 1–9, Sept 2008.
- [4] W. Zhang, D. Meng, Z. Liang, Y. Guo, J. Zhou, and Y. Han, "Research on recognition method of uuv vision based on pso-bp network," in *OCEANS 2015*, pp. 1–6, Oct 2015.
- [5] Z. Wei, G. Yi, M. Detao, L. Zhicheng, and C. Tao, "Research on diving control of underactuated uuv based on model predictive control with artificial bee colony algorithm," in *34th Chinese Control Conference (CCC)*, pp. 4073–4078, July 2015.
- [6] L. De Silva and H. Ekanayake, "Behavior-based robotics and the reactive paradigm: A survey," in *Proceedings of International Workshop in Data Mining and Artificial Intelligence (DMAI '08)*, pp. 36–43, Dec. 2008.
- [7] A. Turing, "Computing machinery and intelligence," *Mind, New Series*, vol. 59, pp. 433–460, Oct. 1950.
- [8] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, 1994.
- [9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.
- [10] M. G. Hinchey, R. Sterritt, and C. Rouff, "Swarms and swarm intelligence," *Computer*, vol. 40, pp. 111–113, April 2007.
- [11] R. Hassan, B. Cohanin, O. de Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proc. 46th AIAA/ASME/AHS/ASC Structures, Structural Dynamics & Materials Conference*, 2005.