HEURISTIC OPTIMIZATION OF BAT ALGORITHM FOR HETEROGENEOUS SWARMS USING PERCEPTION

Sivayazi Kappagantula1*, Saipranav Vojjala1,2, Aditya Arun Iyer1,2, Gurunadh Velidi3,4, Sampath Emani5, Seshu Kumar Vandrangi6

1 Department of Mechatronics, Manipal Institute of Technology, MAHE, Eshwar Nagar, Manipal, 576104, Karnataka, India
2 Department of Computer Science Engineering, Manipal Institute of Technology, MAHE, Eshwar Nagar, Manipal, 576104, Karnataka, India.
3 Department of Aerospace Engineering, University of Petroleum and Energy Studies, India
4 Department of Mechanical Engineering, Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea
5 Individual Collaborator, India
6 Department of Mechanical Engineering, University of Technology PETRONAS, Malaysia

Received: 17 March 2023
Accepted: 26 June 2023
First Online: 11 July 2023

Abstract: In swarm robotics, a group of robots coordinate with each other to solve a problem. Swarm systems can be heterogeneous or homogeneous. Heterogeneous swarms consist of multiple types of robots as opposed to Homogeneous swarms, which are made up of identical robots. There are cases where a Heterogeneous swarm system may consist of multiple Homogeneous swarm systems. Swarm robots can be used for a variety of applications. Swarm robots are majorly used in applications involving the exploration of unknown environments. Swarm systems are dynamic and intelligent. Swarm Intelligence is inspired by naturally occurring swarm systems such as Ant Colony, Bees Hive, or Bats. The Bat Algorithm is a population-based meta-heuristic algorithm for solving continuous optimization problems. In this paper, we study the advantages of fusing the Meta-Heuristic Bat Algorithm with Heuristic Optimization. We have implemented the Meta-Heuristic Bat Algorithm and tested it on a heterogeneous swarm. The same swarm has also been tested by segregating it into different homogeneous swarms by subjecting the heterogeneous swarm to a heuristic optimization.

Keywords: Swarm Intelligence, Bat Algorithm, Heuristic Algorithm, Meta-Heuristic Algorithm, Heterogeneous Swarm System, Homogeneous Swarm System

*Corresponding author: sivayazi.k@manipal.edu (S, Kappagantula)
saipranavvojjala948@gmail.com (S, Vojjala), adityaaruniyer02@gmail.com (A. A. Iyer),
guru.velidi@gmail.com (G. Velidi), Sampath.evs@gmail.com (S, Emani), seshu1353@gmail.com (S. K. Vandrangi)
1. Introduction

Swarm Robots are self-organizing robot systems. Swarm Robot Systems can be characterized as autonomous, dynamic, and intelligent systems. Swarm robots are decentralized, i.e., their sensing and communication capabilities are local. They are collaborative and complete tasks given by cooperating and coordinating with other robots in the same swarm system.

Swarm robots’ behavior can be classified into two significant taxonomies, i.e., Methods and Collective Behaviors.

![Fig. 1 Spatial Behaviors and Methods as explained in Brambilla et al. (2013)]

As seen in the above figure, the collective behaviors of a swarm robot system can be broken down into Spatial Organization, Navigation, and Decision Making. Collective Behaviors are used to automate swarm systems to perform complex tasks. For example, spatially organizing behaviors can be used to organize the robots into a particular shape. Navigation and Decision-Making behaviors can be used to develop collision-free motion planners for swarm systems. These characteristics are discussed in detail in Brambilla et al. (2013). These behaviors are applied in all classes of swarm robots, i.e., aerial, terrestrial, aquatic, and outer space (Schranz et al., 2020), discusses specific swarm research projects and industry products demonstrating Swarm Collective Behaviors.

1.1 Evolution of Swarm Intelligence

For centuries, numerous animal species have benefited from swarm intelligent behavior, a natural phenomenon that has enhanced their chances of survival. The capacity of animal swarms to scatter and explore a vast region, as well as swiftly gather together when a single agent detects food or danger, is well-documented.
Heuristic optimization of bat algorithm for heterogeneous swarms using perception

This phenomenon is known as swarming (Williams, 2018). “Swarm intelligence is defined as a swarm’s use of decentralization and self-organizing behaviors in their decision-making to adapt to change and solve problems”.

Most of the swarm intelligence methods are nature inspired. Swarm intelligent algorithms like the Ant Colony Algorithm, Bees Hive Algorithm, or Particle Swarm Optimization Algorithm were developed by studying naturally occurring swarms (Williams, 2018). Swarm intelligent behaviors can be carried out by two basic search behaviors used in all swarm-based searching algorithms, namely exploration, and exploitation. Exploration refers to searching an unexplored area of the feasible region, while exploitation refers to the intensification of the search in the narrowed down promising regions. The success of these algorithms highly depends on how these two search behaviors are balanced (Tilahun, 2019).

1.2 Bat Algorithm for Swarm Robots

The Bat Algorithm is a population-based meta-heuristic algorithm for solving continuous optimization problems. It has proved to be better than other nature-inspired algorithms (Zebari, Almufti, & Abdulrahman, 2020). It was originally proposed by Prof. Xin- She Yang in 2010 (Yang, 2010). The bat algorithm is based on the echolocation behavior of microbats. The varying emission pulse rates and loudness observed in microbats form the basis of their echolocation behavior. The Meta-Heuristic Bat Algorithm is simple, flexible, and can solve a wide range of problems efficiently. It provides rapid convergence at the initial stages by switching from exploration to exploitation. The Bat algorithm is used to generate multiple trajectory points for a single bot from which the optimal one was selected.

Swarm systems require control algorithms working with path planners to move each robot on its calculated path. In order to design an efficient system, it is important to understand the role of Swarm Control in Swarm Intelligent Systems.

1.3 Swarm Control

Swarm control combines communication and sensing, using information from both to calculate the motions the robots want to make. Swarm robots’ control is decentralized and autonomous because they don’t share a single global controller or have a third-party commander (Tan & Zheng, 2013).

Swarm control can be categorized into two major fields, heterogeneous swarm control and homogeneous swarm control. Conventional robot swarms are homogeneous, i.e., they are made up of entirely identical robots by function and design.

Trying to build a flexible homogeneous swarm of robots presents two problems. The primary issue has to do with the engineering considerations of the robots. It is not realistically feasible to equip each swarm agent with all the necessary features and capabilities to execute every task in the environment. Secondly, strict homogeneity presents a scaling problem (Campbell, 2019). A heterogeneous model can be adopted to solve these issues. But when discrete homogeneous swarms within a heterogeneous swarm are required to combine, it
complicates the swarm control mechanism. In later sections of this paper, we will provide a solution for the same.

Heterogeneous swarms combine different kinds of robots, each with different functionality. A simple heterogeneous swarm for military applications is defined with three kinds of robots: supply unit, defender unit, and attacker unit (McCook & Esposito, 2007).

<table>
<thead>
<tr>
<th>Supply Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Move to goal (defined convoy goal)</td>
</tr>
<tr>
<td>2. Avoid collision with other supply units</td>
</tr>
<tr>
<td>3. Avoid collisions with obstacles</td>
</tr>
<tr>
<td>4. Avoid attacking units</td>
</tr>
<tr>
<td>5. Match average velocity of all supply units in sensor range</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defender Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Move to changing goal (position between supply units and attackers)</td>
</tr>
<tr>
<td>2. Avoid collision with other defender units</td>
</tr>
<tr>
<td>3. Avoid collision with supply units</td>
</tr>
<tr>
<td>4. Avoid collisions with obstacles</td>
</tr>
<tr>
<td>5. Intercept attackers that get too close to the supply units</td>
</tr>
<tr>
<td>6. Match average velocity of all supply units in sensor range</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attacker Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Used for simulation purposes only (not autonomous in real life)</td>
</tr>
<tr>
<td>1. Move to goal (centroid of supply units)</td>
</tr>
<tr>
<td>2. Avoid collisions with obstacles</td>
</tr>
<tr>
<td>3. Match average velocity of all supply units in sensor range</td>
</tr>
</tbody>
</table>

Fig. 2 Attacker, Defender and Supply Units as shown in McCook and Esposito (2007)

From the above figure, we observe that by dividing the heterogeneous swarm into multiple homogeneous swarm systems, we circumvent both the engineering constraints and the scaling issue, as not every agent needs to be equipped to solve all the issues posed by the environment, implying that only a few robots of each type need to be created.

1.4 Application of Swarm Intelligence

There are several uses for swarm robotics in terrestrial, aerial, and aquatic robots. As this paper focuses on Unmanned Ground Vehicles (UGV) swarms, only terrestrial applications, particularly military ones, are examined. In instances where it would be dangerous for soldiers to execute missions, military robots can be used. Although resorting to military force should only be done in extreme circumstances, it is vital to look into technological advancements that might assist in sparing lives on both sides while maintaining the effectiveness of the military force.

1.5 Military Applications of Swarm Robots

Military applications of swarm robots are twofold: Exploratory and Attack. The former has undergone extensive study, inspired the creation of numerous heuristic and meta-heuristic swarm intelligence algorithms, and serves as the primary objective for many other applications, such as unmanned interplanetary missions.
Advantages in Exploratory Application (Suárez, Iglesias, & Gálvez, 2019):

Parallelization: This feature makes the swarm more flexible and allows the swarm members to perform different actions simultaneously at different locations. It makes the system more efficient for complex tasks, as it can be broken down into parts and solved independently.

- Task enablement: Through task enablement, swarm systems can do specific tasks that are impossible or very difficult for a single robot.
- Distributed sensing and action: Swarm intelligence systems make use of their inherited auto-configuration and self-organization capabilities to respond well to rapidly changing environments making them highly adaptive. This also ensures that failure of a single unit does not affect the completion of the given task making it highly robust and giving it a high fault tolerance.
- Scalability: Allows the swarm system to be flexible and easily vary in size without reprogramming the whole swarm.

The latter objective, attack, can be fulfilled by weaponizing the swarm system. This is out of the scope of this paper.

1.6 Example for Military Application of Swarm Systems

Consider a scenario in which a large military convoy crosses potentially dangerous unexplored terrain. In such circumstances, a robot swarm can be sent to explore and map the region. Each agent can be weaponized in case enemy troops are spotted and fire breaks out. In this paper, the primary objective is neither exploration nor attack. Instead, our objective is to program a heterogeneous swarm consisting of different homogeneous swarms, i.e., each homogeneous robot swarm is armed differently or is programmed to perform a different function, as discussed in the earlier section on attacker and defender (McCook & Esposito, 2007). Also, Heuristic Optimizations are widely utilized in various aviation mechanisms as well (Gultepe, 2023; Oo, 2023).

1.7 Challenge of Autonomous Navigation by Robots

All autonomous vehicles are equipped with sensors using which they navigate the environment they are deployed in. The decisions made by the vehicle depend on the sensor data it receives. This particular data processing is what makes autonomous vehicles intelligent systems. The better the data from the sensors are processed, the more optimized the traversal is. Optimizing collision-free path planning is a significant challenge that autonomous robots face. Reaching the target with minimum cost while avoiding collisions with stationary or moving objects is a vast area of ongoing research. The problem of path planning can be further classified into global and local path planning. Global path planners require information regarding the environment to be provided before operation. In comparison, local path planners hardly have any information on the terrain. Local and Global planners are often used together to increase the efficiency of path-planning algorithms (Halder,
1.8 Path Planning in Swarm Robots

Path planning in swarm systems can be either heuristic or meta-heuristic. Heuristic approaches are designed for a specific problem. In contrast, meta-heuristic algorithms are designed to be independent of the problem. "They know nothing about the problem they are applied to; hence, they treat functions as black boxes" (Halder, 2021). Due to its problem-independent design, meta-heuristic algorithms can be applied to many problems and are commonly associated with path planning in swarm systems.

1.9 Requirement of Heuristics in Military Swarms

Metaheuristic algorithms, by nature, do not provide the most optimal solution to a given problem. While in many problems, the most optimized solution is often not required, in certain use cases such as the one considered in this paper, optimization is critical. "Black-box optimization implies that the algorithm has no knowledge of the problem apart from the fact that, given any candidate solution, it can obtain the objective function value of that solution" (Chopard et al., 2018). For a military convoy switching between exploratory and attack states, knowledge of its surroundings is required to calculate the optimal goal positions for the attack formation. This implies that as enemy lines change, these goals should also change. The dynamic changing of these goals is possible only by introducing heuristics.

1.10 Hybrid Algorithm

A hybrid algorithm consists of two or more algorithms collectively and cooperatively solving a predefined problem" (Ting et al., 2015). Hybrid Algorithms can take on different structures depending on the situation. Hybrid algorithms consist of two or more algorithms in which one algorithm is used to locate the optimal parameters for another algorithm. In certain cases, different components of algorithms, like mutation and crossover, can be used to enhance the performance of another algorithm within the hybrid. A hybrid algorithm can achieve the most optimal solution to a problem (Ting et al., 2015).

This paper discusses a hybrid algorithm for path planning in autonomous swarms that is heuristic with respect to the heterogeneous swarm but metaheuristic with respect to each homogeneous swarm. The hybrid used in this paper uses the output generated by the heuristic algorithm as input to the metaheuristic algorithm.

According to Voß (2001), heuristic refers to the process of finding optimal parameters through trial and error. Metaheuristic, on the other hand, refers to a higher level of heuristic that often outperforms simple heuristics (Voß, 2001). Glover (1990) and Glover and Kochenberger (2006) define metaheuristics as a "master strategy" that guides and modifies other heuristics to generate solutions beyond local optimality. Metaheuristics are generally more tolerant to
Heuristic optimization of bat algorithm for heterogeneous swarms using perception changes in external parameters compared to heuristic algorithms. A review of metaheuristics was conducted by Glover (1990), which provides an excellent overview of the field.

The software stack for the proposed novel algorithm leverages the advantages of both heuristic and meta-heuristic approaches to problem-solving by combining heuristic and meta-heuristic stages of path planning for optimal swarm control. The individual bots can be arranged into two distinct swarm clusters: homogeneous and heterogeneous. In homogeneous swarm clusters, all the bots of a common characteristic are grouped into a single cluster. While in a heterogeneous swarm cluster, all the robots behave as a single cluster. Depicted below is the setup used to test the convergence of the algorithm depicting the homogeneous swarm clusters grouped based on color.

![Fig. 3 Attack Formation Points on Convex Hull](image)

The heterogeneous swarm depicts the heuristic nature of the algorithm where the final configuration/shape of the heterogeneous swarm variably changes with respect to parameters such as distance from the nearest node, global position, and goal heading, which are called "stochastic estimates" of the algorithm. The algorithm replaces the homogeneous swarm system by a single characteristic point that can represent the entire homogeneous swarm. The idea is to collapse the homogeneous swarm system to a single point referred to as homogeneous swarm centers in an attempt to simplify the system. Every swarm center is characterized by a specific pose from which we can derive the pose of all the nodes of the respective homogeneous swarm center. Thus the goal of the planning algorithm for the homogeneous swarm centers simplifies arranging n-points to form a specific shape. The positions of these swarm centers are decided by the perception data and visual inferences of object detection obtained from the onboard cameras of the individual swarm nodes. A feature classification module has been developed to detect these objects and estimate their position in the global frame. The positions of the enemy military establishment thus allow us to calculate effective counter-attack formation shape, which in turn decides the goals for the swarm centers. The algorithm proposed in this paper is designed keeping in mind a military convoy scenario in which a heterogeneous swarm breaks into multiple homogeneous swarms upon detecting threats. Since military swarms have two major objectives: Exploratory and Attack, we consider a scenario in which an autonomous military convoy encounters threats while exploring unknown terrain. Upon detecting threats, the swarm would break from the
exploratory state into an attack state. A wedge formation defines the attack state.

2. Methodology

By definition, a global path planner plots the most optimal path between the desired destination and the current position, while the term local planner references obstacle avoidance algorithms. A good path-planning algorithm consists of both a global planner and a local planner. In this paper, we focus on optimizing path planning as a whole, not its individual components. Therefore, although the global and local planners are not explicitly mentioned in the text, it’s understood that the path planner consists of a global and local planner to which the following heuristic and metaheuristic optimizations are made to make the algorithm functional for the use case chosen.
Heuristic optimization of bat algorithm for heterogeneous swarms using perception

2.1 Heuristic Optimisation for Heterogeneous Swarm

This module consists of a perception node and a convex hull generator node hereby referenced as goal generator. The poses of the tanks detected from the perception node are used to calculate the ideal pose for each homogeneous swarm center. These points are calculated using a convex hull algorithm. The goal generator is responsible for generating the ideal pose required for the team leader of each homogeneous swarm in the wedge. The pose for the remaining swarm robots in the homogeneous swarm is calculated relative to the pose of their respective homogeneous swarms’ team leader.

2.2 Perception

The perception node decides the optimal position for the leader of each homogeneous swarm cluster based on feedback from depth cameras mounted to the swarm robots to achieve heuristic behavior. The swarm local clusters are calculated when the perception node classifies the image from the input video stream as a tank.

The feature classification model uses an RCNN-based Markov Chain model (Girshick, 2015) aided by discrimination trees (Zhao, Jiang, & Stathaki, 2017) for quick classification on live image feed. It uses

*Fig. 6 RCNN Model Shape*

input features from the images fed, applying the pre-stored weights and predicting the probability of the existence of a feature along with its pose in the camera frame. Theoretically, the features that the model identifies purely depend on the training data (Krizhevsky, Sutskever, & Hinton, 2017). For our case, image data of tanks have been used to showcase the potential military applications of the algorithm. Thus, the implementation of the classifier enables the algorithm to follow a heuristic approach to swarm behavior while still ensuring flexibility with respect to the use case for the algorithm. The incorporation of perception-based estimation of global goals for the homogeneous swarm central features eliminates the issues concerning graph-based Particle-Swarm Optimisation (PSO) Algorithms, such as the local-minima problem and its inability to tackle solving multi-objective optimization problems. The weights are transmitted to every individual node of the swarm and are used to process the image onboard without sharing it with the rest of the swarm. Data containing the classifier’s output is relayed to every node of the swarm.
planning are computed on the node chosen as the central feature.

The aim is to develop a mathematical model to arrange swarm clusters into a formation governed by the poses of the enemy formation obtained by detecting army establishments via computer vision and deep learning. For the application considered in the paper, we have used army tanks as the enemy establishment and implemented the detection model. This can be scaled to any establishment by controlling the image data fed into the training model described below.

Before turning control over to the heuristic and meta-heuristic controller of the heterogeneous and homogeneous swarms, respectively, the methodology for the object detection subroutine of the software stack has four key objectives to accomplish (Jain, Yerragolla, & Guha, 2019):

• Object Recognition being the primary goal of this subroutine, is to identify objects in the image data stream.

• Image Classification to predict the type of class of an object in the input image. In the paper, we have only one class, i.e., army tanks. However, it can include multiple classes of military establishments and vehicles.

![Object Detection Methodology](Fig. 7 Object Detection Methodology from [24])

• Object Localization is a major subtask indicating the location of the classes in question and creating a bounding box for the classes recognized.

• Object Detection is the final subtask of the perception module to return all the detected classes and their respective pose in a single data structure as input to the heuristic planner for the heterogeneous swarm.

For the implementation, we have chosen to use the Military Tanks Dataset containing images of various battle formations obtained from scraping images from the internet (Jain et al., 2019) to train the model and generate the weights pre-deployment. These weights are then locally stored on every swarm node, accessed iteratively post-deployment, and applied to the input image obtained from the
Heuristic optimization of bat algorithm for heterogeneous swarms using perception onboard depth camera. The weights are generated using a Recurrent Neural Network (RNN) (Ren et al., 2015) architecture for object detection trained using input images of size \((640\times 480\times 3)\) obtained from the dataset without any scaling as it allows the use of ROI (Region-of-Interest) Pooling (Sermanet et al., 2013) and Spatial Resolution techniques (Elmer, 2006) to improve feature extraction and classification.

The model first converts the input image of size \((640\times 480\times 3)\) into a tensor of 8 feature maps using the Convolution 2D Scaling (Conv2D). This step is repeated three more times to produce 16, 32, and 64 feature maps, respectively. Each tensor index has dimensions \((6\times 6, 64)\) flattened into a single-dimensional vector of shape \((2304)\). The activation function chosen for each layer is Leaky Relu (Girshick et al., 2014) to implement Stochastic Gradient Descent (Ho & Wookey, 2019) based on Binary Cross-Entropy (Oberman & Prazeres, 2019) for loss estimation. An effective learning rate was chosen for fast convergence using the method described in (Chee & Toulis, 2018). The tensors are then convoluted using a Dense layer three times into tensors of shape \((120\times 120), (84\times 84), (10\times 10)\), including a softmax layer that aids feature identification (Alabassy, Safar, & El-Kharashi, 2020; Hu et al., 2018). Finally, the features extracted are filtered to include only the classes we require with a Dense convolutional layer of dimensions \((1,1)\) as the output of the Recurrent Neural Network. The output of the Deep Learning module is a bounding box of all the instances of the output class detected, along with their poses.

The model was compiled using TensorFlow and Keras modules and optimized with Keras’s in-built Adam Optimiser achieving an overall accuracy of 96.235%
on the test.

![Flattening of RCNN](image1)

**Fig. 9 Flattening of RCNN**

![Tank Detection dataset and 94.76% accuracy on an image stream at 50Hz from an Intel Realsense depth camera.](image2)

**Fig. 10 Tank Detection dataset and 94.76% accuracy on an image stream at 50Hz from an Intel Realsense depth camera.**

![Model Summary](image3)

**Fig. 11 Accuracy comparison between CNN and RCNN**
Heuristic optimization of bat algorithm for heterogeneous swarms using perception

**Algorithm 1** Detecting Army tanks via Camera Feed

### Result:
Publish ROS topics containing Poses of army tanks detected
- *images* list of ROS Image messages
- *CurrentPose* list of Poses corresponding to nodes in the heterogeneous swarm

#### while *Iter* lesser than *max iterations* do

#### for *bat* in *bats* do

#### img Subscribe to *bat/camera/image* raw
Append *img* to *images*.

#### end

#### if size of *images* greater than 0 then

*CurrentPose* Object Detection using CNN on *images*
Publish *CurrentPose* as a ROS topic

else
continue
end

2.3 Generation of Convex Hull

The data received from the perception node is unpacked and processed by the planner. The existence of this node is to provide the heuristic nature of the algorithm. It optimizes the traversal using the heuristics obtained from the perception node. Additionally, it condenses the homogeneous swarms into a single point designated as their central swarm center. It provides the goal pose for these centers calculated using geometrical extrapolation equations with the poses from the perception module as input.

The dependency of the shape and alignment of homogeneous swarm centers on the input poses of the enemy line allows the algorithm to optimally compute the arrangement of the individual homogeneous swarm centers to cover the maximum area of overlap, thus making the heterogeneous swarm control heuristic.

**Algorithm 2** Calculating goals for the Homogeneous Swarm using Convex Hull

### Result:
Goals for the homogeneous swarm clusters published on ROS topics
- *GoalPoints* list of Coordinates (x,y)
- *CurrentPose* List of ROS Odometry messages

#### while *Iter* lesser than *max iterations* do

#### pose Subscribe to /tank pose topics
*CurrentPose[i]* pose→*normalpoints*
List of points of intersection of normals on the convex hull ConvexHull
Convex Hull Equation passing through tank positions

#### for *points* in *CurrentPose* do

#### normalEqn[i] Equations of normals to the convex hull at tank positions

#### intercepts[i] Points of intersection of normal equations and convex hull

#### for *point* in *intercepts* do

#### if *point* not in *CurrentPose* then
Append *point* to *GoalPoints*

else
end

#### if size of *GoalPoints* greater than 0 then
Publish *GoalPoints[i]* as a ROS topic

else
continue
end
Extensive work has been done in shape formation in swarm systems, as demonstrated in (Rubenstein, Cornejo, & Nagpal, 2014; Wang & Rubenstein, 2020). In the approach presented in (Rubenstein et al., 2014), robots are unaware of their global position and collectively construct a coordinate system from gradient values and distance information. This coordinate system is used to localize their position and move into the required formation.

Another approach to programming shape assemblies to swarm systems exists in which the algorithm forms the goal shape using two modules: the goal selector and the motion planner (Wang & Rubenstein, 2020). The goal selector chooses a goal from a set of target points given as input using local information. Once a valid goal is picked, the algorithm uses the motion planner node to move toward the goal.

From the above-mentioned papers, we can infer that in swarm systems, shape assembly requires a motion planner and a goal selector which selects points using a local coordinate system. In this paper, we use a meta-heuristic motion planning algorithm optimized with a heuristic goal selector.

### 2.4 Meta-Heuristic Planner for Homogeneous Swarm

Meta-Heuristic algorithms such as Bat Algorithm and Particle Swarm Optimization are often associated with optimizing path planning in swarm robots. These optimizations are beneficial in swarms when navigating to a goal position in an unknown environment. In this paper, we discuss an implementation of the Bat Algorithm for metaheuristic planning in homogenous swarms mainly due to its ability to solve multi-objective optimization problems (Xin-She, 2011; Yang & He, 2013), among other applications, which is a crucial characteristic required by military swarms in order to arrange themselves into various shapes before carrying out an attack.

---

**Algorithm 3** Bat Algorithm as given by Yang (2010)

**Result:** Optimization of Robot Traversal Objective function $f(x)$, $x = (x_1, \ldots, x_d)^T$

1. Initialize the bat population $x_i (i=1,2,\ldots,n)$ and $v_i$
2. Define pulse frequency $f_i$ at $x_i$
3. Initialize pulse rates $r_i$ and the loudness $A_i$
4. while $t <$ Max number of iterations do
   5. if $(\text{rand} > r_i)$ then
      6. Select a solution among the best solutions Generate a local solution around the selected best solution
   7. end
   8. Generate a new solution by flying randomly if $(\text{rand} < A_i f(x) < f(x^*))$ then Accept the new solutions Increase $r_i$ and reduce $A_i$
   9. end
10. Postprocess results and visualization

---
Heuristic optimization of bat algorithm for heterogeneous swarms using perception

This particular implementation of the Metaheuristic Bat Algorithm uses the homo-geneous swarm centers generated by the heuristic planning algorithm as input. The goal pose of the swarm center, while acting as the destination coordinates for the central node in a homogeneous swarm, also determines the goal pose for other bots in its homogeneous swarm meta-heuristically. The wedged shape of each homogeneous swarm is determined by the separation that each agent must maintain from their respective central node. The Bat Algorithm is used with Algorithm 4 for accurate path planning of each swarm agent and quick convergence of each homogeneous swarm.

**Algorithm 4** Planner for each individual swarm node

**Result**: Publish ROS topics containing velocity commands for each swarm node

$Velocity$ Output: velocity for a node $GoalPosition$ global goal for Homogeneous Swarm centre $Reigons$ list of regions segmented from the LaserScan message

```plaintext
while Iter less than max iterations do
    odom Subscribe→ to Odometry message of each swarm node
    position, orientation ← odom.orientation goal ← orientation along z
    axis, lasermsg Subscribe to LaserScan
    reigons ← {min(reigons[left]), min(reigons[centre]), min(reigons[right])}
    Call service decision if distance between GoalPosition and position is lesser than distanceError then
        if reigons[i] lesser than 0.7 then
            $Velocity$ Set appropriate velocity to avoid obstacle Publish $Velocity$ to ROS topic /cmd vel
        else
            if Difference between yaw and orientation greater than OriError then
                $Velocity$ Set appropriate velocity to correct orientation Publish $Velocity$ to ROS topic /cmd vel
            else
                $Velocity$.angular 0 ←
            end
        end
        $Velocity$ Set appropriate linear velocity as a function of distance error Publish $Velocity$ to ROS topic /cmd vel
    endelse
end
```

3. Results

The algorithm has been implemented using the Robot Operating System (ROS) to show proof of concept. The results have been simulated on the Gazebo physics engine. For ease of design, cylindrical-shaped differential drive robots
have been used to represent each swarm agent in the simulation. Due to computational restrictions, the algorithm has been tested for a heterogeneous swarm consisting of 15 bots belonging to 3 homogeneous swarms represented by different colors.

Heuristic optimization algorithms are often evaluated based on their convergence. However, comparing the efficiency of an algorithm directly to others based on this characteristic alone is not enough. To assess the performance of an algorithm, a better approach is to study the distribution of the number of steps needed to achieve convergence to the optimal state (Sun et al., 2012).

The merit of each algorithm has been evaluated based on its ability to converge the 3 homogeneous swarms into the wedge formation. The algorithms have been evaluated by comparing the time it takes for each algorithm to converge each homogeneous swarm system into a wedge and the mean convergence percentage across iterations. In Figure [11], we depict the ideal output, i.e., the complete convergence of 3 homogeneous swarms. Figure [12] shows examples of failed test cases obtained during testing where the algorithm either failed to converge or converged incorrectly.
Heuristic optimization of bat algorithm for heterogeneous swarms using perception

The real-time in tables 1 and 2 represents the total time taken by the heterogeneous swarm to converge completely and is calculated using the real-time factor (RTF) along with the simulation time. Table 1 and 2 consists of RTF, simulation time, and real-time convergence from 30 test simulations. The number of iterations represents the number of times the objective function is called during swarm convergence. It also contains information about the success of convergence, and the overall percentage of cases converged that is updated every iteration.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Real Time Factor</th>
<th>Simulation Time</th>
<th>Real-Time Iterations</th>
<th>Convergence</th>
<th>Sim Time (seconds)</th>
<th>Real-Time Convergence</th>
<th>Convergence (Percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.26</td>
<td>7:00.00</td>
<td>4:14.00</td>
<td>82127</td>
<td>420</td>
<td>254</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.68</td>
<td>14:39.18</td>
<td>23:28.46</td>
<td>171324</td>
<td>incomplete</td>
<td>879.18</td>
<td>1408.464</td>
</tr>
<tr>
<td>3</td>
<td>0.73</td>
<td>6:44.32</td>
<td>1:34.03</td>
<td>55389</td>
<td>complete</td>
<td>404.342</td>
<td>94.034</td>
</tr>
<tr>
<td>4</td>
<td>0.73</td>
<td>6:39.512</td>
<td>1:33.187</td>
<td>54837</td>
<td>complete</td>
<td>399.512</td>
<td>93.187</td>
</tr>
<tr>
<td>5</td>
<td>0.67</td>
<td>6:52.526</td>
<td>1:52.392</td>
<td>67674</td>
<td>complete</td>
<td>412.526</td>
<td>112.392</td>
</tr>
<tr>
<td>6</td>
<td>0.61</td>
<td>7:12.02</td>
<td>2:38.612</td>
<td>86852</td>
<td>incomplete</td>
<td>432.02</td>
<td>158.612</td>
</tr>
<tr>
<td>7</td>
<td>0.72</td>
<td>7:06.773</td>
<td>2:03.658</td>
<td>81565</td>
<td>complete</td>
<td>426.773</td>
<td>123.658</td>
</tr>
<tr>
<td>8</td>
<td>0.65</td>
<td>6:53.057</td>
<td>1:47.937</td>
<td>67869</td>
<td>complete</td>
<td>413.057</td>
<td>107.937</td>
</tr>
<tr>
<td>9</td>
<td>0.67</td>
<td>7:05.873</td>
<td>2:21.406</td>
<td>80701</td>
<td>complete</td>
<td>425.873</td>
<td>141.406</td>
</tr>
<tr>
<td>10</td>
<td>0.71</td>
<td>6:59.320</td>
<td>2:08.98</td>
<td>74152</td>
<td>complete</td>
<td>419.32</td>
<td>128.98</td>
</tr>
<tr>
<td>11</td>
<td>0.64</td>
<td>7:26.803</td>
<td>2:56.44</td>
<td>101625</td>
<td>complete</td>
<td>446.803</td>
<td>176.44</td>
</tr>
<tr>
<td>12</td>
<td>0.7</td>
<td>7:17.990</td>
<td>2:30.550</td>
<td>92822</td>
<td>complete</td>
<td>437.99</td>
<td>150.55</td>
</tr>
<tr>
<td>13</td>
<td>0.74</td>
<td>6:39.570</td>
<td>1:29.833</td>
<td>54402</td>
<td>incomplete</td>
<td>399.57</td>
<td>89.833</td>
</tr>
<tr>
<td>14</td>
<td>0.7</td>
<td>6:59.240</td>
<td>2:03.470</td>
<td>74072</td>
<td>incomplete</td>
<td>419.24</td>
<td>123.47</td>
</tr>
<tr>
<td>15</td>
<td>0.73</td>
<td>6:40.505</td>
<td>1:34.696</td>
<td>55337</td>
<td>incomplete</td>
<td>400.505</td>
<td>94.696</td>
</tr>
<tr>
<td>16</td>
<td>0.71</td>
<td>6:41.359</td>
<td>1:36.783</td>
<td>56191</td>
<td>complete</td>
<td>401.359</td>
<td>96.783</td>
</tr>
<tr>
<td>17</td>
<td>0.64</td>
<td>6:41.23</td>
<td>1:36.986</td>
<td>55855</td>
<td>complete</td>
<td>401.23</td>
<td>96.986</td>
</tr>
<tr>
<td>18</td>
<td>0.64</td>
<td>6:39.974</td>
<td>1:34.787</td>
<td>54806</td>
<td>complete</td>
<td>399.974</td>
<td>94.787</td>
</tr>
<tr>
<td>19</td>
<td>0.64</td>
<td>7:31.692</td>
<td>1:51.276</td>
<td>66524</td>
<td>complete</td>
<td>451.692</td>
<td>111.276</td>
</tr>
<tr>
<td>20</td>
<td>0.64</td>
<td>6:43.730</td>
<td>1:40.835</td>
<td>58562</td>
<td>complete</td>
<td>403.73</td>
<td>100.835</td>
</tr>
<tr>
<td>21</td>
<td>0.69</td>
<td>06:51.46</td>
<td>01:47.03</td>
<td>51778</td>
<td>complete</td>
<td>411.46</td>
<td>107.03</td>
</tr>
<tr>
<td>22</td>
<td>0.67</td>
<td>6:38.776</td>
<td>1:38.795</td>
<td>53608</td>
<td>complete</td>
<td>398.776</td>
<td>98.795</td>
</tr>
<tr>
<td>23</td>
<td>0.61</td>
<td>7:04.297</td>
<td>2:09.648</td>
<td>79129</td>
<td>complete</td>
<td>424.297</td>
<td>129.648</td>
</tr>
<tr>
<td>24</td>
<td>0.7</td>
<td>6:48.084</td>
<td>1:48.280</td>
<td>62916</td>
<td>incomplete</td>
<td>408.084</td>
<td>108.28</td>
</tr>
<tr>
<td>25</td>
<td>0.68</td>
<td>6:44.750</td>
<td>1:41.198</td>
<td>59582</td>
<td>complete</td>
<td>404.75</td>
<td>101.198</td>
</tr>
<tr>
<td>26</td>
<td>0.72</td>
<td>6:49.616</td>
<td>1:50.450</td>
<td>64448</td>
<td>incomplete</td>
<td>409.616</td>
<td>110.45</td>
</tr>
<tr>
<td>27</td>
<td>0.66</td>
<td>6:41.002</td>
<td>1:43.397</td>
<td>55834</td>
<td>incomplete</td>
<td>401.002</td>
<td>103.397</td>
</tr>
<tr>
<td>28</td>
<td>0.69</td>
<td>6:42.445</td>
<td>1:42.779</td>
<td>54277</td>
<td>complete</td>
<td>402.445</td>
<td>102.779</td>
</tr>
<tr>
<td>29</td>
<td>0.69</td>
<td>6:36.996</td>
<td>1:31.891</td>
<td>51828</td>
<td>complete</td>
<td>396.996</td>
<td>91.891</td>
</tr>
<tr>
<td>30</td>
<td>0.67</td>
<td>6:51.777</td>
<td>1:56.837</td>
<td>66009</td>
<td>incomplete</td>
<td>411.777</td>
<td>116.837</td>
</tr>
</tbody>
</table>

Table 1: Statistics and Time Analysis for Pure Meta-Heuristic Bat Algorithm
Table 2: Statistics and Time Analysis for Meta-Heuristic Bat Algorithm with Heuristic Optimization

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Real Time Factor</th>
<th>Simulation Time (seconds)</th>
<th>Real-Time Iterations</th>
<th>Convergence</th>
<th>Sim Time (seconds)</th>
<th>Real-Time (seconds)</th>
<th>Convergence (Percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.64</td>
<td>6:37.842</td>
<td>52674</td>
<td>complete</td>
<td>397.842</td>
<td>90.172</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.68</td>
<td>6:45.546</td>
<td>58777</td>
<td>complete</td>
<td>405.546</td>
<td>105.991</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.63</td>
<td>6:43.494</td>
<td>58326</td>
<td>complete</td>
<td>403.494</td>
<td>105.825</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.72</td>
<td>6:35.779</td>
<td>50205</td>
<td>complete</td>
<td>395.779</td>
<td>90.205</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.65</td>
<td>6:35.490</td>
<td>50322</td>
<td>complete</td>
<td>395.490</td>
<td>95.052</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.74</td>
<td>6:36.593</td>
<td>51425</td>
<td>complete</td>
<td>396.593</td>
<td>92.894</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0.6</td>
<td>6:34.484</td>
<td>49316</td>
<td>complete</td>
<td>394.484</td>
<td>89.776</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.56</td>
<td>6:36.639</td>
<td>51471</td>
<td>complete</td>
<td>396.639</td>
<td>98.189</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0.6</td>
<td>6:36.642</td>
<td>51474</td>
<td>complete</td>
<td>396.642</td>
<td>96.528</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.66</td>
<td>6:42.008</td>
<td>56840</td>
<td>complete</td>
<td>402.008</td>
<td>103.776</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0.64</td>
<td>6:50.264</td>
<td>68356</td>
<td>complete</td>
<td>410.264</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0.66</td>
<td>6:36.415</td>
<td>51247</td>
<td>complete</td>
<td>396.415</td>
<td>91.861</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>0.69</td>
<td>6:45.883</td>
<td>60715</td>
<td>complete</td>
<td>405.883</td>
<td>104.785</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0.66</td>
<td>6:44.592</td>
<td>59424</td>
<td>complete</td>
<td>404.592</td>
<td>105.594</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0.68</td>
<td>6:36.423</td>
<td>51255</td>
<td>complete</td>
<td>396.423</td>
<td>93.533</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>0.61</td>
<td>6:38.067</td>
<td>52899</td>
<td>complete</td>
<td>398.067</td>
<td>106.27</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>0.64</td>
<td>6:41.830</td>
<td>56662</td>
<td>complete</td>
<td>401.83</td>
<td>106.462</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>0.56</td>
<td>6:23.766</td>
<td>38598</td>
<td>incomplete</td>
<td>383.766</td>
<td>74.042</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0.66</td>
<td>6:47.298</td>
<td>62120</td>
<td>complete</td>
<td>407.298</td>
<td>110.934</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>0.62</td>
<td>6:38.552</td>
<td>53384</td>
<td>complete</td>
<td>398.552</td>
<td>97.318</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>0.62</td>
<td>6:42.787</td>
<td>57601</td>
<td>complete</td>
<td>402.787</td>
<td>100.857</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>0.72</td>
<td>6:44.394</td>
<td>59266</td>
<td>complete</td>
<td>404.394</td>
<td>103.51</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>0.67</td>
<td>6:41.691</td>
<td>56523</td>
<td>complete</td>
<td>401.691</td>
<td>101.024</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>0.68</td>
<td>6:43.419</td>
<td>58251</td>
<td>complete</td>
<td>403.419</td>
<td>105.36</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>0.66</td>
<td>6:47.100</td>
<td>65302</td>
<td>complete</td>
<td>407.1</td>
<td>117.824</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>0.65</td>
<td>6:44.785</td>
<td>59617</td>
<td>complete</td>
<td>404.785</td>
<td>114.068</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>0.67</td>
<td>6:39.161</td>
<td>53993</td>
<td>complete</td>
<td>399.161</td>
<td>98.701</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>0.65</td>
<td>6:53.149</td>
<td>67981</td>
<td>complete</td>
<td>413.149</td>
<td>123.771</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>0.58</td>
<td>6:24.743</td>
<td>40241</td>
<td>incomplete</td>
<td>384.743</td>
<td>75.221</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0.65</td>
<td>7:06.259</td>
<td>81091</td>
<td>complete</td>
<td>426.259</td>
<td>146.801</td>
<td>1</td>
</tr>
</tbody>
</table>

Assessing the efficiency of an algorithm often involves evaluating its convergence rate. While some research has been done to analyze the convergence rate of population-based random optimization algorithms, it remains an open problem for arbitrary objective functions due to the mathematical complexity involved (Sun et al., 2012). The convergence rate of an algorithm is defined as the state when all swarm nodes successfully transition from the exploratory to the
Heuristic optimization of bat algorithm for heterogeneous swarms using perception attack state or vice versa, as mentioned in the literature.

The performance of both algorithms for the aforementioned parameters is graphically depicted below:

![Fig. 14 Comparison of Simulation Time over Iterations](image1)

![Fig. 15 Comparison of Real-Time over Iterations](image2)

![Fig. 16 Comparison of Convergence over Iterations](image3)

4. Discussion

We can conclude that the Meta-Heuristic Bat Algorithm with Heuristic Optimization is efficient in practical applications via Figure 14’s demonstration of a mean R2 score within the limits of the Shifted Sphere evaluation criteria function as seen in the CEC2005 benchmark suite (Suganthan et al., 2005). A semi-theoretical analysis of the time complexity and convergence rate of the Meta-Heuristic Bat Algorithm with Heuristic Optimization shows that it has superior convergence qualities and lower computational complexity in comparison to the Pure Meta-Heuristic Bat Algorithm. The algorithm’s global convergence is
sufficiently proved in Figure 15 to support this claim.

### Table 3: Summary of Observations

<table>
<thead>
<tr>
<th>Algorithm Type</th>
<th>Mean Simulation Time (Seconds)</th>
<th>Mean Real-Time (Seconds)</th>
<th>Mean Convergence Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure Meta-Heuristic Bat Algorithm</td>
<td>428.7966333</td>
<td>160.9543667</td>
<td>70</td>
</tr>
<tr>
<td>Meta-Heuristic Bat Algorithm with Heuristic Optimization</td>
<td>401.1698333</td>
<td>102.2114667</td>
<td>93.33333333</td>
</tr>
</tbody>
</table>

### 2.5 Shortcomings

Discussed below are some of the generally observed open perspectives \((\text{Gad, 2022})\) and future research directions with respect to shortcomings that can be drawn from the work presented in the paper.

**Premature convergence** According to the literature \((\text{Xu, Wu, & Jiang, 2015; Yuan & Yin, 2014})\), when using the traditional Meta-Heuristic Swarm Optimization algorithm, the convergence toward a locally optimal solution occurs when the search is initiated through random initial conditions. This convergence can lead to deceptive optimization results, where both the individual particle optimum and the group/global optimum converge towards the local optimal solution, thereby failing to ensure the discovery of the global optimal solution. Consequently, the fast convergence capability of the algorithm is rendered ineffective. In the future, research could incorporate random sampling of control parameters \((\text{Sun, Song, & Chen, 2019})\), stability analysis \((\text{Bonyadi & Michalewicz, 2015})\), and redistributing mechanisms \((\text{Qi, Ju, & Xu, 2018})\).

**High-dimensional search space** Heuristic Swarm Optimization has gained significant attention for its effectiveness in classifying high-dimensional data despite the curse of the dimensionality problem \((\text{Ali Yahya, 2018})\). Although recent studies have demonstrated the effectiveness of Heuristic Swarm Optimization-based feature selection, the challenge remains in applying it to datasets with tens of thousands of features \((\text{Tran, Zhang, & Xue, 2016})\) due to the large search space. However, the Heuristic Swarm Optimization algorithm can be adapted to solve high-dimensional feature selection problems by selecting only a small set of relevant features \((\text{Gu, Cheng, & Jin, 2018})\), leading to similar or even better classification accuracy. In this area, novel approaches such as Monte Carlo \((\text{Beskos et al., 2017})\) methods have been proposed to minimize the number of chosen features and maximize classification accuracy in Heuristic Swarm Optimization applications.

**Memory requirement** All swarm systems possess memory as an essential feature. In the context of Heuristic Swarm Optimization, it is valuable to investigate the potential beneficial role of historical memory from an evolutionary perspective.

This is because Heuristic Swarm Optimization benefits from explicit or implicit historical memory by storing promising solutions and reusing them in later
Heuristic optimization of bat algorithm for heterogeneous swarms using perception stages (Li et al., 2015), thus enhancing the search process (Hino et al., 2016). Moreover, using historical memory to generate a new inertia weight through a parameter adaptation mechanism presents an opportunity to further improve Heuristic Swarm Optimization (Li, 2018). Hence, it is imperative to address the challenge of improving Heuristic Swarm Optimization using historical memory (Luo et al., 2016), and exploring ways to adaptively set the memory size can also be an interesting topic for future research.

**Parameter selection** Effective determination of control parameters is crucial for achieving the best performance in Heuristic Swarm Optimization-based algorithms. However, it can be challenging to guide the process of parameter selection. To overcome this challenge, future research in this area could involve selecting the best parameters through simulations (Cui et al., 2017), conducting parametric analysis with limited computational resources (Serani et al., 2016), and employing heuristics (Lorenzo et al., 2017) to perform hyperparameter selection.

### 2.6 Future Scope

In this paper, we demonstrate the algorithm for a specific military scenario. However, the algorithm can theoretically be extended to a wide array of varying applications by simply changing the data set from which the object detection model is trained and the heuristics are calculated. Similarly, The metaheuristic algorithm can be extended to arrange the homogeneous swarms into different formations according to the application’s requirements.

Apart from the military application mentioned in the paper, the proposed optimization would be particularly useful in autonomous spacecraft navigation. As more and more spacecraft are sent into space, the need for autonomous navigation becomes increasingly important. The Heuristic Optimisation proposed could be used to help spacecraft autonomously navigate through the complex and unpredictable space environment.

Additionally, Swarm robots equipped with this algorithm could explore abandoned mines and search for valuable minerals. In the exploratory state, robots can map out the mine while in the attack state, they can defend against potential threats or obstacles that could harm the mission.

### 4. Conflict of Interest

No conflict of interest exists.

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

### 5. Author Contributions

- Author 1: Sivayazi Kappagantula Conceived and designed the analysis Collected the data. Contributed data or analysis tools Wrote the paper.

- Author 2: Saipranav Vojjala Conceived and designed the analysis. Contributed data or analysis tools. Wrote the paper. Performed the analysis.
- Author 3: Aditya Arun Iyer Conceived and designed the analysis. Contributed data or analysis tools. Wrote the paper. Performed the analysis.

**Funding**

No funding was received for this work.

**Acknowledgments**

Not applicable

**Ethics approval**

Not applicable

**Consent to participate.**

Not applicable

**Consent for publication**

Not applicable

**References**


Heuristic optimization of bat algorithm for heterogeneous swarms using perception


https://doi.org/10.1504/IJBIC.2011.042259
https://doi.org/10.1504/IJWMC.2015.074034
https://doi.org/10.1007/978-3-642-12538-6_6
https://doi.org/10.1504/IJBIC.2013.055093
https://doi.org/10.1109/TAC.2014.2381454
http://dx.doi.org/10.14419/ijsw.v8i1.30120
https://doi.org/10.1155/2017/7186120