Operational Research in Engineering Sciences: Theory and Applications
Vol. 6, Issue 2, 2023, pp. 52-77
ISSN: 2620-1607
eISSN: 2620-1747
DOI: https://doi.org/10.31181/oresta/060204

HEURISTIC OPTIMIZATION OF BAT ALGORITHM FOR HETEROGENEOUS SWARMS USING PERCEPTION

Sivayazi Kappagantula[1*], Saipranav Vojjala[1,2], Aditya Arun Iyer[1,2], Gurunadh Velidi[3,4], Sampath Emani[5], Seshu Kumar Vandrangi[6]

[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

Research Paper

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.

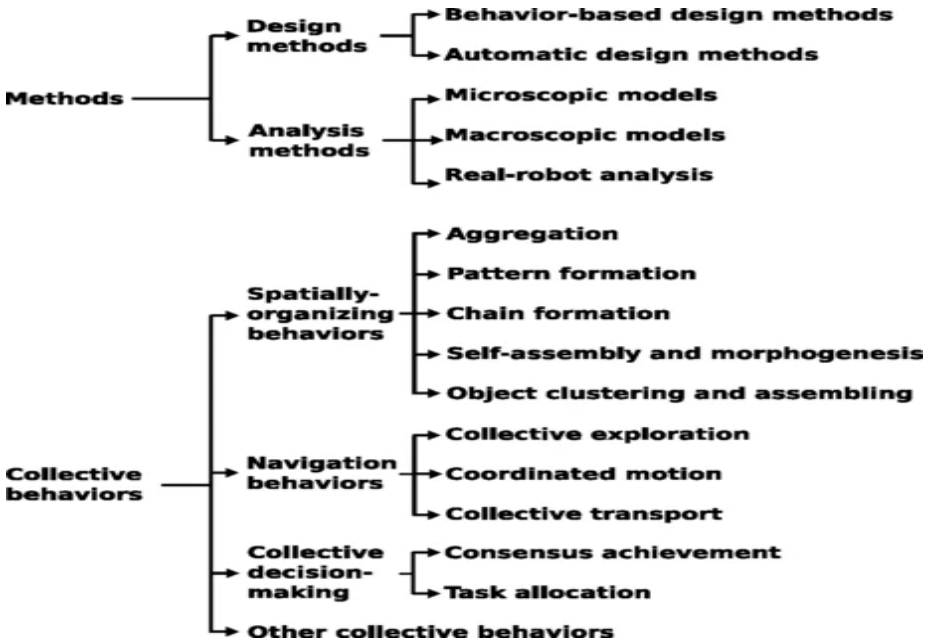


Fig. 1 Spatial Behaviors and Methods as explained in Brambilla et al. (2013)

As seen in the above figure, the collective behaviors of a swarm robot systemcan 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 andindustry products demonstrating Swarm Collective Behaviors.

1.1 Evolution of Swarm Intelligence

For centuries, numerous animal species have benefited from swarm intelligent behav- ior, a natural phenomenon that has enhanced their chances of survival. The capacity ofanimal 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.

This phenomenon is

known as swarming (Williams, 2018). "Swarm intelligence is defined as a swarm's use of decentral- ization 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 tothe 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 con-tinuous 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 generatemultiple 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 importantto 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 decentralizedand 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 con- trol 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 isnot 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 func-tionality. A simple heterogeneous swarm for military applications is defined with threekinds of robots: supply unit, defender unit, and attacker unit (McCook & Esposito, 2007).

Supply Unit	1. Move to goal (defined convoy goal) 2. Avoid collision with other supply units 3. Avoid collisions with obstacles 4. Avoid attacking units 5. Match average velocity of all supply units in sensor range
Defender Unit	1. Move to changing goal (position between supply units and attackers) 2. Avoid collision with other defender units 3. Avoid collision with supply units 4. Avoid collisions with obstacles 5. Intercept attackers that get too close to the supply units 6. Match average velocity of all supply units in sensor range
Attacker Unit	*Used for simulation purposes only (not autonomous in real life) 1. Move to goal (centroid of supply units) 2. Avoid collisions with obstacles 3. Match average velocity of all supply units in sensor range

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 constraintsand 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 wouldbe 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 makesthe system more efficient for complex tasks, as it can be broken down into partsand 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 inher- ited 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. Thisis 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 signifi-cant 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 environ- ment 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,

2021).

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 problemthey 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, optimizationis critical. "Black-box optimization implies that the algorithm has no knowledge ofthe 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 betweenexploratory 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 meta-heuristic 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 solu-tions beyond local optimality. Metaheuristics are generally more tolerant to

changesin 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 advantagesof both heuristic and meta-heuristic approaches to problem-solving by combiningheuristic 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 characteris- tic 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 conver- gence of the algorithm depicting the homogenous swarm clusters grouped based on color.

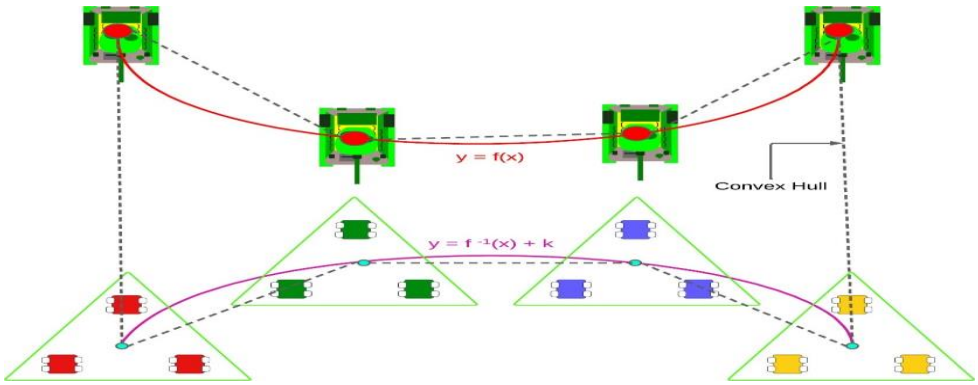
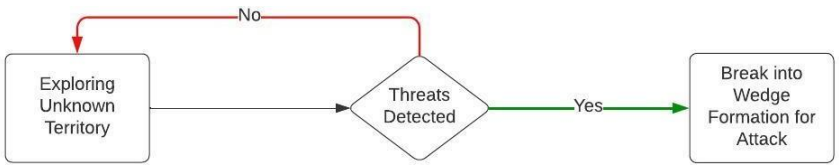


Fig. 3 Attack Formation Points on Convex Hull

The heterogeneous swarm depicts the heuristic nature of the algorithm where the final configuration/shape of the heterogeneous swarm variably changes with respect toparameters 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 hasbeen 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 effectivecounter-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 swarmsupon 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 wouldbreak from the

exploratory state into an attack state. A wedge formation defines the attack state.

Fig. 4 Basic Flow of Algorithm. Definition of States



The wedge is a common attack state. It is a fire-team formation where soldiersare spaced about 10 meters apart. The team leader moves at the point of the wedge. The team leader can opt to disengage the wedge formation by closing up the distance between the soldiers when enemy contact is not likely. This approach facilitates more straightforward command and control over the fire team. The fire team leader expandsthe wedge formation by increasing the distance between soldiers in the event of prob- able enemy contact. This increases the protection and security of the fire team from enemy contact. "The wedge is easy to control, is flexible, provides good security, and allows the team members to fire immediately in all directions" (Hackworth & England, 2003; United States Department of the Army, 1977; United States Government US Army, 2007, 2019).

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 obsta- cle 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 plan- ners 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 metaheuristicoptimizations are made to make the algorithm functional for the use case chosen.

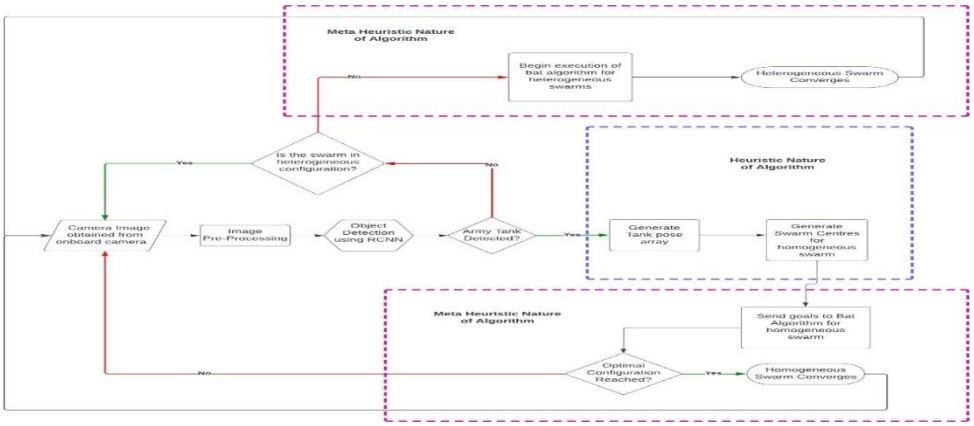


Fig. 5 Algorithmic Design Architecture Flowchart

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 responsiblefor generating the ideal pose required for the team leader of each homogeneous swarmin 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 homogeneousswarm cluster based on feedback from depth cameras mounted to the swarm robots to achieve heuristic behavior. The swarm local clusters are calculated when the perceptionnode 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

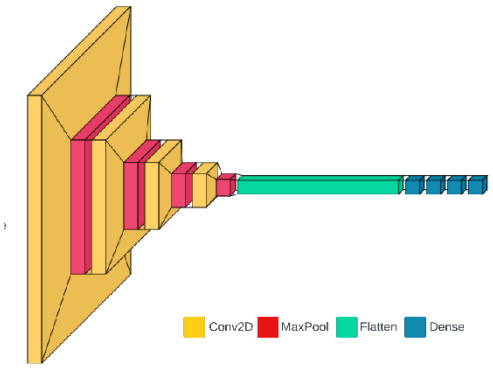


Fig. 6 RCNN Model Shape

input features from the images fed, applying the pre-stored weights and predictingthe 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 trainingdata (Krizhevsky, Sutskever, & Hinton, 2017). For our case, image data of tanks have been used to showcase the poten-tial 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 incorporationof perception-based estimation of global goals for the homogeneous swarm central fea-tures 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 nodeof 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. The calculations for the

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 objectsin 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 multipleclasses of military establishments and vehicles.

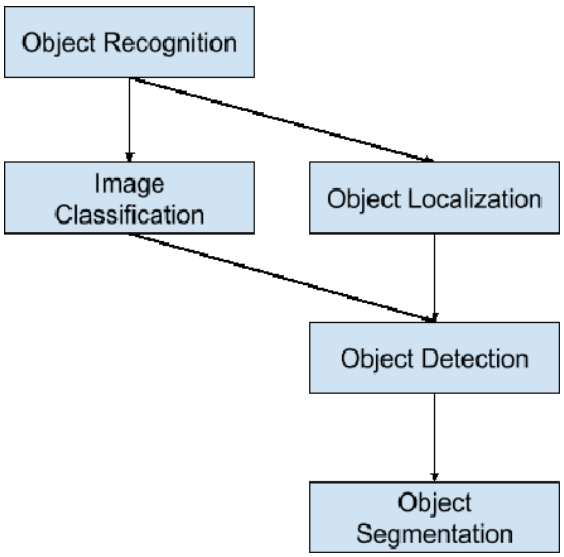


Fig. 7 Object Detection Methodology from [24]

- 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 theheuristic 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

onboard depth camera. The weights are generated using a Recurrent Neural Network(RNN) (Ren et al., 2015) architec-ture for object detection trained using input images of size (640px,480px,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.

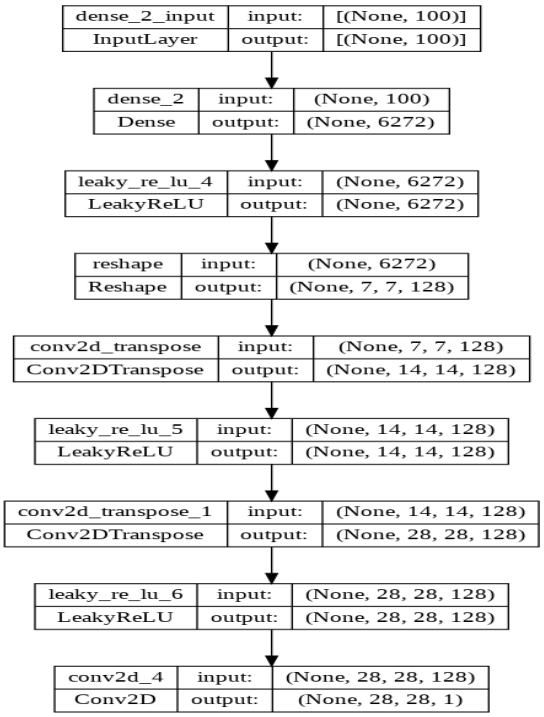


Fig. 8 RCNN Model Layers

The model first converts the input image of size (640 px,480 px,3) into a tensor of 8feature 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 hasdimensions (6px,6px,64) flattened into a single-dimensional vector of shape (,2304 px). The activation function chosen for each layer is Leaky Relu (Girshick et al., 2014) to implement Stochas- tic 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 px), (,84 px),(,10 px), including a softmax layer that aids feature identi- fication (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) as the output of the Recurrent Neural Network. The output of the Deep Learning module is a boundingbox 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.

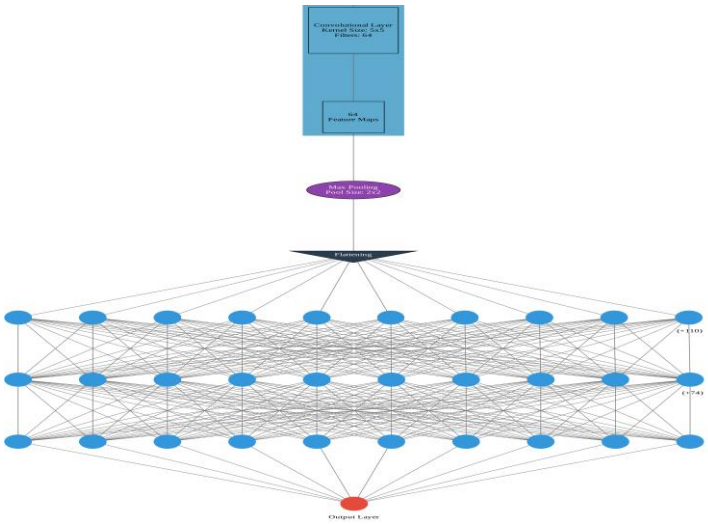


Fig. 9 Flattening of RCNN

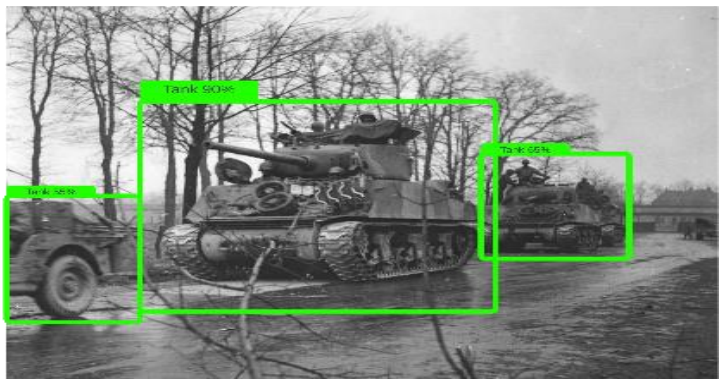


Fig. 10 Tank Detection dataset and 94.76% accuracy on an image stream at 50Hz from an Intel Realsense depth camera.

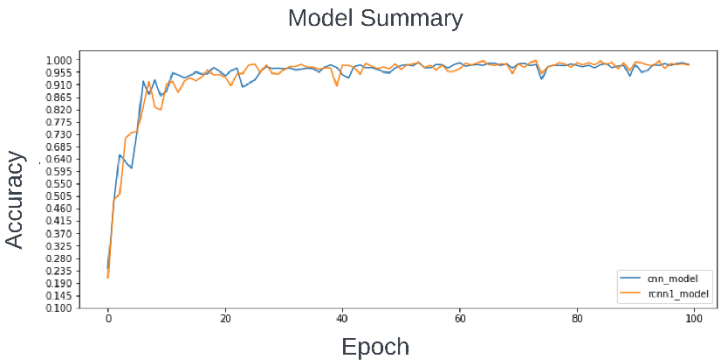


Fig. 11 Accuracy comparison between CNN and RCNN

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 *bats* list of 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
endend

2.3 Generation of Convex Hull

The data received from the perception node is unpacked and processed by the plan- ner. 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 usinggeometrical 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 areaof 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 Odometrymessages **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* Con← vex 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*
elseend
end
if *size of GoalPoints greater than 0* **then**
Publish *GoalPoints*[*i*] as a ROS topic
else
continue
endend

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 globalposition and collectively construct a coordinate system from gradient values and dis- tance 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 andthe 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 usinga 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 Optimizationare often associated with optimizing path planning in swarm robots. These optimiza- tions 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 variousshapes before carrying out an attack.

Algorithm 3 Bat Algorithm as given by Yang (2010)

Result: Optimization of Robot Traversal Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_d)^T$
Initialize the bat population $\mathbf{x}_i (i = 1, 2, \ldots, n)$ and $\mathbf{v}_i$
Define pulse frequency f_i at $\mathbf{x}_i$
Initialize pulse rates r_i and the loudness A_i
while $t <$ *Max number of iterations* **do**
Generate new solutions by adjusting frequency and updating velocities and locations/solutions [equations (2) to (4)]
if $(rand > r_i)$ **then**
Select a solution among the best solutions Generate a local solution around the selected best solution
end
Generate a new solution by flying randomly **if** ($rand < A_i \& f(\mathbf{x}_i) < f(\mathbf{x}_*)$) **then**Accept the new solutions Increase r_i and reduce A_i
end
Rank the bats and find the current best $\mathbf{x}_*$**end**
Postprocess results and visualization

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 cen-tral 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
V elocity Output velocity for a node *GoalPosition* global goal for Homogeneous Swarm centre *Reigons* list of regions segmented from the LaserScan message
while *Iter lesser than max iterations* **do**
odom Subscribe to *Odometry* message of each swarm node
position *odom.position* *orientation* *odom.orientation* *goal*
yaw Euler angle of orientation along *z*
axis *lasermsg* Subscribe to *LaserScan* {
message of each swarm} node
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**
V elocity Set appropriate velocity to avoid obstacle Publish *V elocity* to ROS topic */cmd vel*
else
if *Difference between yaw and orientation greater than OriError* **then**
V elocity Set appropriate velocity to correct orientation Publish *V elocity* to ROS topic */cmd vel*
else
V elocity.angular 0
end
V elocity Set appropriate *linear velocity* as a function of distance error Publish *V elocity* to ROS topic */cmd vel*
endelse
V elocity Zero velocity Publish *V elocity* to ROS topic */cmd vel*
endend

3. Results

The algorithm has been implemented using the Robot Operating System (ROS) toshow 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 rep- resent 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 bet- ter 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 the3 homogeneous swarms into the wedge formation. The algorithms have been evaluatedby 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 wherethe algorithm either failed to converge or converged incorrectly.

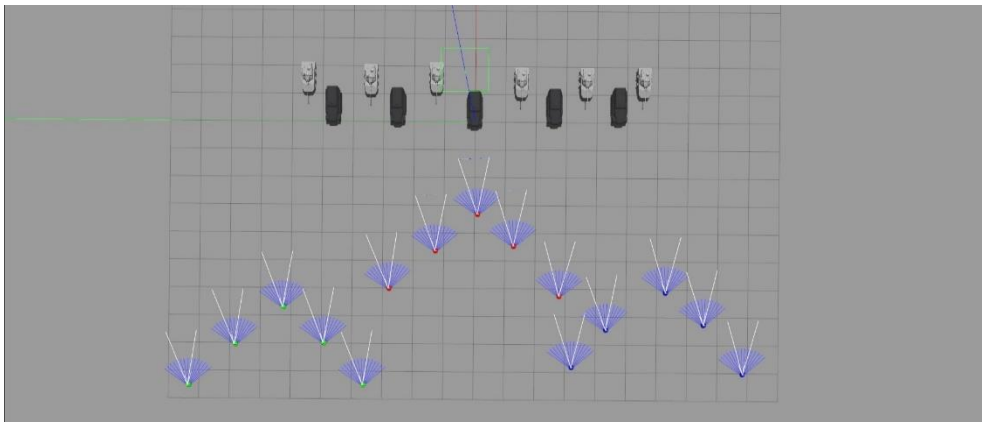


Fig. 12 Case of Complete Convergence for all Homogeneous Swarms

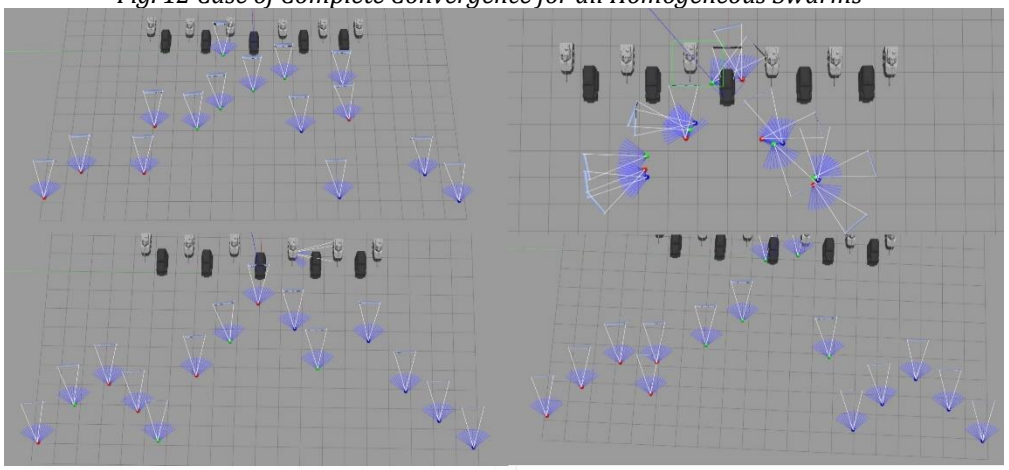


Fig. 13 Cases of Incomplete & Incorrect Convergence encountered during testing.

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 conver- gence. It also contains information about the success of convergence, and the overall percentage of cases converged that is updated every iteration.

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

Test Case	Real Time Factor	Simulation Time	Real-Time	Iterations	Convergence	Sim Time (seconds)	Real-Time (seconds)	Convergence	Convergence (Percentage)
1	0.26	7: 00.00	4: 14.00	82127	complete	420	254	1	100
2	0.68	14: 39.180	23:28.464	171324	incomplete	879.18	1408.464	0	50
3	0.73	6: 44.342	1: 34.034	55389	complete	404.342	94.034	1	66.66666667
4	0.73	6:39.512	1: 33.187	54837	complete	399.512	93.187	1	75
5	0.67	6: 52.526	1: 52.392	67674	complete	412.526	112.392	1	80
6	0.61	7: 12.020	2: 38.612	86852	incomplete	432.02	158.612	0	66.66666667
7	0.72	7:06.773	2:03.658	81565	complete	426.773	123.658	1	71.42857143
8	0.65	6:53.057	1: 47.937	67869	complete	413.057	107.937	1	75
9	0.67	7:05.873	2: 21.406	80701	complete	425.873	141.406	1	77.77777778
10	0.71	6:59.320	2:08.98	74152	complete	419.32	128.98	1	80
11	0.64	7: 26.803	2: 56.44	101625	complete	446.803	176.44	1	81.81818182
12	0.7	7: 17.990	2: 30.550	92822	complete	437.99	150.55	1	83.33333333
13	0.74	6: 39.570	1: 29.833	54402	incomplete	399.57	89.833	0	76.92307692
14	0.7	6:59.240	2:03.470	74072	incomplete	419.24	123.47	0	71.42857143
15	0.73	6: 40.505	1: 34.696	55337	incomplete	400.505	94.696	0	66.66666667
16	0.71	6: 41.359	1: 36.783	56191	complete	401.359	96.783	1	68.75
17	0.64	6: 41.23	1: 36.986	55855	complete	401.23	96.986	1	70.58823529
18	0.64	6:39.974	1: 34.787	54806	complete	399.974	94.787	1	72.22222222
19	0.64	7:31.692	1: 51.276	66524	complete	451.692	111.276	1	73.68421053
20	0.64	6: 43.730	1: 40.835	58562	complete	403.73	100.835	1	75
21	0.69	06: 51.46	01: 47.03	51778	complete	411.46	107.03	1	76.19047619
22	0.67	6: 38.776	1: 38.795	53608	complete	398.776	98.795	1	77.27272727
23	0.61	7:04.297	2:09.648	79129	complete	424.297	129.648	1	78.26086957
24	0.7	6: 48.084	1: 48.280	62916	incomplete	408.084	108.28	0	75
25	0.68	6:44.750	1: 41.198	59582	complete	404.75	101.198	1	76
26	0.72	6: 49.616	1: 50.450	64448	incomplete	409.616	110.45	0	73.07692308
27	0.66	6: 41.002	1: 43.397	55834	incomplete	401.002	103.397	0	70.37037037
28	0.69	6: 42.445	1, 42.779	54277	complete	402.445	102.779	1	71.42857143
29	0.69	6: 36.996	1: 31.891	51828	complete	396.996	91.891	1	72.4137931
30	0.67	6: 51.777	1: 56.837	66009	incomplete	411.777	116.837	0	70

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

Test Case	Real Time Factor	Simulation Time	Real-Time	Iterations	Convergence	Sim Time (seconds)	Real-Time (seconds)	Convergence	Convergence (Percentage)
1	0.64	6: 37.842	1.30 .172	52674	complete	397.842	90.172	1	100
2	0.68	6: 45.546	1: 45.991	58777	complete	405.546	105.991	1	100
3	0.63	6: 43.494	1: 45.825	58326	complete	403.494	105.825	1	100
4	0.72	6:35.779	1: 30.205	50205	complete	395.779	90.205	1	100
5	0.65	6: 35.490	1: 35.052	50322	complete	395.49	95.052	1	100
6	0.74	6:36.593	1: 32.894	51425	complete	396.593	92.894	1	100
7	0.6	6: 34.484	1: 29.776	49316	complete	394.484	89.776	1	100
8	0.56	6: 36.639	1: 38.189	51471	complete	396.639	98.189	1	100
9	0.6	6: 36.642	1: 36.528	51474	complete	396.642	96.528	1	100
10	0.66	6: 42.008	1: 43.776	56840	complete	402.008	103.776	1	100
11	0.64	6: 50.264	2: 00.00	68356	complete	410.264	120	1	100
12	0.66	6: 36.415	1: 31.861	51247	complete	396.415	91.861	1	100
13	0.69	6: 45.883	1: 44.785	60715	complete	405.883	104.785	1	100
14	0.66	6:44.592	1:45.594	59424	complete	404.592	105.594	1	100
15	0.68	6: 36.423	1: 33.533	51255	complete	396.423	93.533	1	100
16	0.61	6:38.067	01: 46.27	52899	complete	398.067	106.27	1	100
17	0.64	6: 41.830	1: 46.462	56662	complete	401.83	106.462	1	100
18	0.56	6: 23.766	1: 14.042	38598	incomplete	383.766	74.042	0	94.44444444
19	0.66	6:47.298	1: 50.934	62120	complete	407.298	110.934	1	94.73684211
20	0.62	6: 38.552	1: 37.318	53384	complete	398.552	97.318	1	95
21	0.62	6: 42.787	1: 40.857	57601	complete	402.787	100.857	1	95.23809524
22	0.72	6:44.394	1: 43.510	59266	complete	404.394	103.51	1	95.45454545
23	0.67	6: 41.691	1: 41.024	56523	complete	401.691	101.024	1	95.65217391
24	0.68	6: 43.419	1: 45.360	58251	complete	403.419	105.36	1	95.83333333
25	0.66	6:47.100	1: 57.824	65302	complete	407.1	117.824	1	96
26	0.65	6: 44.785	1: 54.068	59617	complete	404.785	114.068	1	96.15384615
27	0.67	6: 39.161	1: 38.701	53993	complete	399.161	98.701	1	96.2962963
28	0.65	6:53.149	2:03.771	67981	complete	413.149	123.771	1	96.42857143
29	0.58	6:24.743	1: 15.221	40241	incomplete	384.743	75.221	0	93.10344828
30	0.65	7: 06.259	2: 26.801	81091	complete	426.259	146.801	1	93.33333333

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 conver- gence rate of an algorithm is defined as the state when all swarm nodes successfully transition from the exploratory to the

attack state or vice versa, as mentioned in the literature.

The performance of both algorithms for the aforementioned parameters is graphi-cally depicted below:

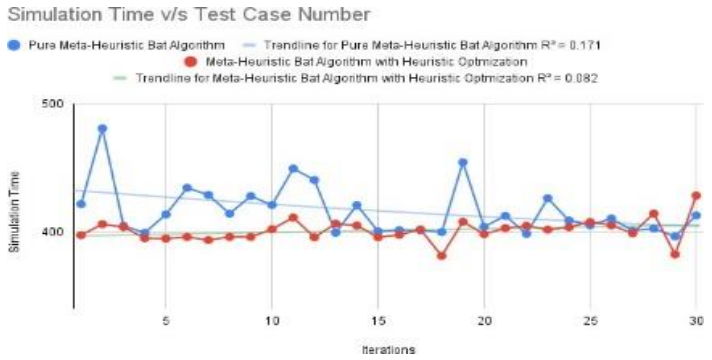


Fig. 14 Comparison of Simulation Time over Iterations

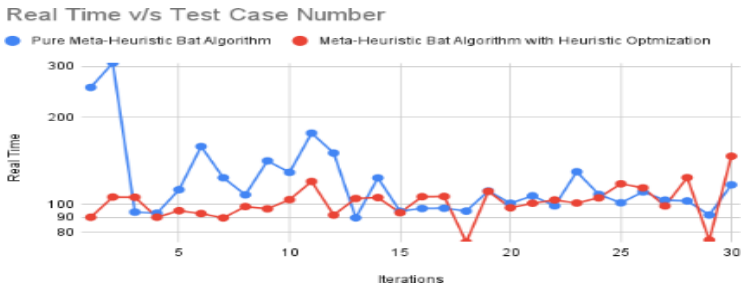


Fig. 15 Comparison of Real-Time over Iterations

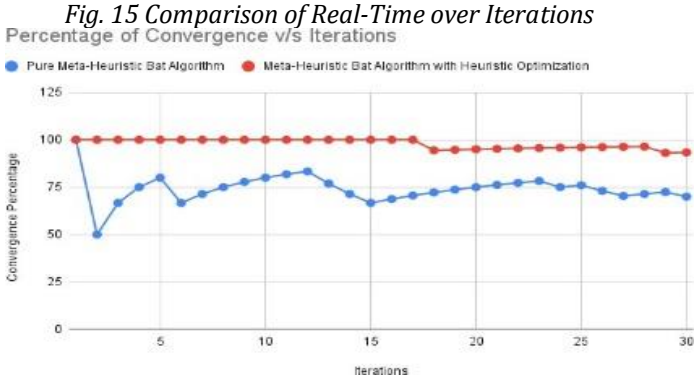


Fig. 16 Comparison of Convergence over Iterations

4. Discussion

We can conclude that the Meta-Heuristic Bat Algorithm with Heuristic Optimizationis 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 theCEC2005 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 Optimizationshows 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

	Mean Simulation Time (Seconds)	Mean Real-Time (Seconds)	Mean Convergence Percentage (%)
Pure Meta-Heuristic Bat Algorithm	428.7966333	160.9543667	70
Meta-Heuristic Bat Algorithm with Heuristic Optimization	401.1698333	102.2114667	93.33333333

2.5 Shortcomings

Discussed below are some of the generally observed open perspectives (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 (Xu, Wu, & Jiang, 2015; Yuan & Yin, 2014), when using thetraditional 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 (Sun, Song, & Chen, 2019),stability analysis (Bonyadi & Michalewicz, 2015), and redistributing mechanisms (Qi, Ju, & Xu, 2018).

High-dimensional search space Heuristic Swarm Optimization has gained sig- nificant attention for its effectiveness in classifying high-dimensional data despite the curse of the dimensionality problem (Ali Yahya, 2018). Although recent studies have demonstrated the effectiveness of Heuristic Swarm Optimization-based feature selection, the chal- lenge remains in applying it to datasets with tens of thousands of features (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 smallset of relevant features (Gu, Cheng, & Jin, 2018), leading to similar or even better classification accuracy.In this area, novel approaches such as Monte Carlo (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 fea- ture. 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 his- torical memory by storing promising solutions and reusing them in later

stages (Li et al., 2015), thus enhancing the search process (Hino et al., 2016). Moreover, using historical memory to generatea new inertia weight through a parameter adaptation mechanism presents an oppor- tunity to further improve Heuristic Swarm Optimization (Li, 2018). Hence, it is imperativeto 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 application sce- nario. 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 canbe 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 opti- mization would be particularly useful in autonomous spacecraft navigation. As more and more spacecraft are sent into space, the need for autonomous navigation becomesincreasingly 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 mapout 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 withthis publication and there has been no significant financial support for this workthat could have influenced its outcome.

5. Author Contributions

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

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

Alabassy, B., Safar, M., & El-Kharashi, M. W. (2020). A high-accuracy implementation for softmax layer in deep neural networks. In 2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS) (pp. 1-6). IEEE. https://doi.org/10.1109/DTIS48698.2020.9081313

Ali Yahya, A. (2018). Centroid particle swarm optimisation for high-dimensional data classification. Journal of Experimental & Theoretical Artificial Intelligence, 30(6), 857-886. https://doi.org/10.1080/0952813X.2018.1509378

Beskos, A., Crisan, D., Jasra, A., Kamatani, K., & Zhou, Y. (2017). A stable particle filter for a class of high-dimensional state-space models. Advances in Applied Probability, 49(1), 24-48. https://doi.org/10.1017/apr.2016.77

Bonyadi, M. R., & Michalewicz, Z. (2015). Stability analysis of the particle swarm optimization without stagnation assumption. IEEE Transactions on Evolutionary Computation, 20(5), 814-819. https://doi.org/10.1109/TEVC.2015.2508101

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence, 7(1), 1-41. https://doi.org/10.1007/s11721-012-0075-2
Campbell, B. J. (2019). Autonomous cueing within heterogeneous robot swarms. (Doctoral dissertation, Monterey, CA; Naval Postgraduate School). https://apps.dtic.mil/sti/citations/trecms/AD1080168
Chee, J., & Toulis, P. (2018). Convergence diagnostics for stochastic gradient descent with constant learning rate. In International Conference on Artificial Intelligence and Statistics (pp. 1476-1485). PMLR. http://proceedings.mlr.press/v84/chee18a.html
Chopard, B., Tomassini, M., Chopard, B., & Tomassini, M. (2018). Performance and limitations of metaheuristics. In An Introduction to Metaheuristics for Optimization (pp. 191-203). Springer, Cham. https://doi.org/10.1007/978-3-319-93073-2_11
Cui, H., Shu, M., Song, M., & Wang, Y. (2017). Parameter selection and performance comparison of particle swarm optimization in sensor networks localization. Sensors, 17(3), 487. https://doi.org/10.3390/s17030487
Elmer, P. (2006). Spatial Resolution in ATR-FT-IR imaging: measurement and interpretation. Technical Note. https://resources.perkinelmer.com/corporate/cmsresources/images/44-74872tch_spatialresolutioninatrft-irimaging.pdf
Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. Archives of computational methods in engineering, 29(5), 2531-2561. https://doi.org/10.1007/s11831-021-09694-4
Girshick, R. (2015). Fast R-CNN. In 2015 IEEE International Conference on Computer Vision (ICCV) (pp. 1440-1448). IEEE. https://doi.org/10.1109/ICCV.2015.169
Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In 2014 IEEE Conference on Computer Vision and Pattern Recognition (pp. 580-587). IEEE. https://doi.org/10.1109/CVPR.2014.81
Glover, F. (1990). Tabu search: A tutorial. Interfaces, 20(4), 74-94. https://doi.org/10.1287/inte.20.4.74
Glover, F. W., & Kochenberger, G. A. (2006). Handbook of metaheuristics (Vol. 57). Springer Science & Business Media. https://doi.org/10.1007/b101874
Gu, S., Cheng, R., & Jin, Y. (2018). Feature selection for high-dimensional classification using a competitive swarm optimizer. Soft Computing, 22, 811-822. https://doi.org/10.1007/s00500-016-2385-6
Gultepe, I. (2023). A Review on Weather Impact on Aviation Operations: Visibility, Wind, Precipitation, Icing. Journal of Airline Operations and Aviation Management, 2(1), 1-44. https://doi.org/10.56801/jaoam.v2i1.1
Hackworth, D. H., & England, E. (2003). Steel My Soldiers' Hearts: The Hopeless to Hardcore Transformation of US Army, 4th Battalion, 39th Infantry, Vietnam. Simon and Schuster. https://www.simonandschuster.com/books/Steel-My-Soldiers-Hearts/David-H-Hackworth/9780743246132
Halder, R. K. (2021). Particle swarm optimization in global path planning for swarm of robots. In Applying Particle Swarm Optimization: New Solutions and Cases for Optimized Portfolios (pp. 209-232). Springer. https://doi.org/10.1007/978-3-030-70281-6_12
Hino, T., Ito, S., Liu, T., & Maeda, M. (2016). Set-based particle swarm optimization with status memory for knapsack problem. Artificial Life and Robotics, 21, 98-105. https://doi.org/10.1007/s10015-015-0253-6
Ho, Y., & Wookey, S. (2019). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. IEEE access, 8, 4806-4813.

https://doi.org/10.1109/ACCESS.2019.2962617
Hu, R., Tian, B., Yin, S., & Wei, S. (2018). Optimization of softmax layer in deep neural network using integral stochastic computation. Journal of Low Power Electronics, 14(4), 475-480. https://doi.org/10.1166/jolpe.2018.1579
Jain, N., Yerragolla, S., & Guha, T. (2019). Performance Analysis of Object Detection and Tracking Algorithms for Traffic Surveillance Applications using Neural Networks. In 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 690-696). IEEE. https://doi.org/10.1109/I-SMAC47947.2019.9032502
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90. https://doi.org/10.1145/3065386
Li, J., Zhang, J., Jiang, C., & Zhou, M. (2015). Composite particle swarm optimizer with historical memory for function optimization. IEEE transactions on cybernetics, 45(10), 2350-2363. https://doi.org/10.1109/TCYB.2015.2424836
Li, W. (2018). Improving particle swarm optimization based on neighborhood and historical memory for training multi-layer perceptron. Information, 9(1), 16. https://doi.org/10.3390/info9010016
Lorenzo, P. R., Nalepa, J., Ramos, L. S., & Pastor, J. R. (2017). Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. In Proceedings of the genetic and evolutionary computation conference companion (pp. 1864-1871). https://doi.org/10.1145/3067695.3084211
Luo, W., Sun, J., Bu, C., & Liang, H. (2016). Species-based particle swarm optimizer enhanced by memory for dynamic optimization. Applied Soft Computing, 47, 130-140. https://doi.org/10.1016/j.asoc.2016.05.032
McCook, C. J., & Esposito, J. M. (2007). Flocking for heterogeneous robot swarms: A military convoy scenario. In 2007 Thirty-Ninth Southeastern Symposium on System Theory (pp. 26-31). IEEE. https://doi.org/10.1109/SSST.2007.352311
Oberman, A. M., & Prazeres, M. (2019). Stochastic Gradient Descent with Polyak's Learning Rate. arXiv preprint arXiv:1903.08688. https://arxiv.org/abs/1903.08688
Oo, K. T. (2023). Review of the History of Mesoscale Convective System Forecasts on Aviation. Journal of Airline Operations and Aviation Management, 2(1), 68-85. https://doi.org/10.56801/jaoam.v2i1.4
Qi, X., Ju, G., & Xu, S. (2018). Efficient solution to the stagnation problem of the particle swarm optimization algorithm for phase diversity. Applied optics, 57(11), 2747-2757. https://doi.org/10.1364/AO.57.002747
Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031
Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. Science, 345(6198), 795-799. https://doi.org/10.1126/science.1254295
Schranz, M., Umlauft, M., Sende, M., & Elmenreich, W. (2020). Swarm robotic behaviors and current applications. Frontiers in Robotics and AI, 7, 36. https://doi.org/10.3389/frobt.2020.00036
Serani, A., Leotardi, C., Iemma, U., Campana, E. F., Fasano, G., & Diez, M. (2016). Parameter selection in synchronous and asynchronous deterministic particle swarm

optimization for ship hydrodynamics problems. Applied Soft Computing, 49, 313-334. https://doi.org/10.1016/j.asoc.2016.08.028
Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229. https://arxiv.org/abs/1312.6229
Suárez, P., Iglesias, A., & Gálvez, A. (2019). Make robots be bats: specializing robotic swarms to the Bat algorithm. Swarm and Evolutionary Computation, 44, 113-129. https://doi.org/10.1016/j.swevo.2018.01.005
Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Natural Computing, 341-357. https://www.researchgate.net/publication/235710019
Sun, J., Wu, X., Palade, V., Fang, W., Lai, C.-H., & Xu, W. (2012). Convergence analysis and improvements of quantum-behaved particle swarm optimization. Information Sciences, 193, 81-103. https://doi.org/10.1016/j.ins.2012.01.005
Sun, L., Song, X., & Chen, T. (2019). An improved convergence particle swarm optimization algorithm with random sampling of control parameters. Journal of Control Science and Engineering, 2019, 1-11. https://doi.org/10.1155/2019/7478498
Tan, Y., & Zheng, Z.-y. (2013). Research advance in swarm robotics. Defence Technology, 9(1), 18-39. https://doi.org/10.1016/j.dt.2013.03.001
Tilahun, S. L. (2019). Balancing the degree of exploration and exploitation of swarm intelligence using parallel computing. International Journal on Artificial Intelligence Tools, 28(03), 1950014. https://doi.org/10.1142/S0218213019500143
Ting, T., Yang, X.-S., Cheng, S., & Huang, K. (2015). Hybrid metaheuristic algorithms: past, present, and future. Recent advances in swarm intelligence and evolutionary computation, 71-83. https://doi.org/10.1007/978-3-319-13826-8_4
Tran, B., Zhang, M., & Xue, B. (2016). A PSO based hybrid feature selection algorithm for high-dimensional classification. In 2016 IEEE congress on evolutionary computation (CEC) (pp. 3801-3808). IEEE. https://doi.org/10.1109/CEC.2016.7744271
United States Department of the Army. (1977). Soldier's manual of common tasks. Department of Defense], Department of the Army, Headquarters. https://books.google.co.in/books?id=dA_QXI31fo0C
United States Government US Army. (2007). Field Manual FM 3-21.8 (FM 7-8) The Infantry Rifle Platoon and Squad March 2007. CreateSpace Independent Publishing Platform. https://books.google.co.in/books?id=fKyBtgAACAAJ
United States Government US Army. (2019). Field Manual FM 1-02. 1 Operational Terms November 2019. Independently Published. https://books.google.co.in/books?id=-gakzAEACAAJ
Voß, S. (2001). Meta-heuristics: The State of the Art. In A. Nareyek (Ed.), Local Search for Planning and Scheduling (pp. 1-23). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45612-0_1
Wang, H., & Rubenstein, M. (2020). Shape formation in homogeneous swarms using local task swapping. IEEE Transactions on Robotics, 36(3), 597-612. https://doi.org/10.1109/TRO.2020.2967656
Williams, S. M. (2018). Swarm Weapons: Demonstrating a Swarm Intelligent Algorithm for Parallel Attack. US Army School for Advanced Military Studies Fort Leavenworth United States. https://apps.dtic.mil/sti/pdfs/AD1071535.pdf
Xin-She, Y. (2011). Bat algorithm for multi-objective optimization. International Journal of Bio-Inspired Computation, 3(5), 267-274.

https://doi.org/10.1504/IJBIC.2011.042259
Xu, G., Wu, Z.-H., & Jiang, M.-Z. (2015). Premature convergence of standard particle swarm optimisation algorithm based on Markov chain analysis. International Journal of Wireless and Mobile Computing, 9(4), 377-382. https://doi.org/10.1504/IJWMC.2015.074034
Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74). Springer. https://doi.org/10.1007/978-3-642-12538-6_6
Yang, X.-S., & He, X. (2013). Bat algorithm: literature review and applications. International Journal of Bio-inspired computation, 5(3), 141-149. https://doi.org/10.1504/IJBIC.2013.055093
Yuan, Q., & Yin, G. (2014). Analyzing convergence and rates of convergence of particle swarm optimization algorithms using stochastic approximation methods. IEEE Transactions on Automatic Control, 60(7), 1760-1773. https://doi.org/10.1109/TAC.2014.2381454
Zebari, A. Y., Almufti, S. M., & Abdulrahman, C. M. (2020). Bat algorithm (BA): review, applications and modifications. Int J Sci World, 8(1), 1-7. http://dx.doi.org/10.14419/ijsw.v8i1.30120
Zhao, X., Jiang, Y., & Stathaki, T. (2017). Automatic target recognition strategy for synthetic aperture radar images based on combined discrimination trees. Computational Intelligence and Neuroscience, 2017, 7186120. https://doi.org/10.1155/2017/7186120