Prioritized Foraging using Robot Swarms

Nicolaas J. Taljaard and Andries P. Engelbrecht

Department of Computer Science, University of Pretoria, South Africa

This empirical study aims to improve on the prioritized foraging problem through using robotic swarms. The attempt will be introducing clustering before the foraging process. It focuses on swarm intelligence in cooperation with swarm robotics with social insect behaviors as the working force. Previous works shows foraging using a honey-bee approach is most successful in the dynamic environments used. The ant cluster is chosen for no in communication or pheromones are required. It was found that cluster beforehand has potential although it comes at a significantly higher iteration cost.

1 Introduction

Swarm Robotics is an entity constructed of simple behavior robots in collaboration, the inspiration for these robot algorithms draw from social insects such as: Bees, ants and termites [Robots-Dorigo]. These insects are reliant on water, food and organization as a source of survival through search, retrieval and sorting. The described processes are the basis of the algorithms in the form of foraging and clustering [Proposal 6].

This is ideal for the mining industry for ore retrieval with shorting waiting times between blasting through the robotic abilities [Jade]. Honey bees have to explore away from their hive to search and retrieve of resources as the foraging [Proposal 11]. Also clustering which is modeled after how ant organize their dead dependent on the density of corpses locations [Proposal 8]. In mining this will improve efficiency of extracting the gold from out the mines with obstructions and rubble [Jade].

Foraging is commonly used as a benchmark tool due to its complex and for the social level coordination required [Winfield]. As by Lumer, the ant clustering algorithm has been used to solve problems in fields such as cryptography, sorting and graphs [Lumer]. Prioritization foraging as tested by Jade [Jade] has found that the honey bee proves to be more successful in foraging in dynamic environments. This paper introduces clustering before the foraging process to way it effects on the foraging success rate, productivity and increased time requirement.

2 Background

The following selections introduces further information on the field and also the clustering and foraging algorithms that are experimented with.

2.1 Swarm Intelligence / Robotics

Swarm Intelligence (SI) as a method of problem solving through the use of agents interacting with their environment as the source of their knowledge [Beni-Robot]. The agents are represented as robots in a collective body known as Swarm Robotics. As collective intelligence of the swarm it raises the social behavior an individual robot, the test algorithms used are independent of an controlling entity to provide guidance to other robots. The independence of this allows robots to base decision on their immediate environment, their current robotic status and ability.

Swarm-based robotics (SR) is defined by Bonabeau *et al* [Robot-Swarm] that it may be able to perform tasks without needing a explicit representation of the environment. These robots are modeled after insect colonies respectively as desert ants for clustering and honey bees for foraging. Each colony has their own internal communication method to achieving their goals and enable the process that is mimicked on the robots.

SI and SR as a self-organized social insects use a communication method call stigmergy [Robot-Swarm], it can be in a form of sematectonic and sign-based communication. Where sign-based is in the form of physical information exchange and sematectonic by means of environment changes. Stigmergy will be used as follow: Direct where the honey bee broadcasts a foraging site by dancing when it arrives at the hive. Also by in-direct for ant cemetery clustering where each ant exams the environment to affect its decision making.

2.2 Desert-ant Cemetery Clustering

The desert ant is used to model the ant clustering process, it is chosen for these ants are independent of any direct communication or information sharing. Desert ants do not require any pheromones or beacons to provide navigation it only uses vision of its immediate environment [12 Jade]. The cemetery cluster is based on the ant that move there died into clusters dependent on the observed environment and the density of object clusters [77 AI].

As for movement decisions it uses random navigation or if detection movement is towards the lowest density [13 Jade]. Once an object is picked up it is carries towards the highest dens array until it satisfies the dropping probability where it is then dropped. This clustering process will aim at improving the performance of foraging due to the closer group of the same priority objects. The abilities of a desert ant is simplistic and does not require many detectors to be added to the robot making it viable and can be cost effective as only a camera is required.

2.3 Honey-bee Foraging

The honey-bee is proven by Jade [Jade] to be success foraging algorithm when working with a wider variety of grid configurations and object placing. These bees have three roles assigned to them [Jade 7]: Employed-, unemployed foragers and scouts. Scouts search the environments, once an object successfully foraged depending on its desirability it can dance for the unemployed bees. These bees wait for a dance and employed bees are actively foraging a location. As suggested by Jasen [Jade 14] waiting bees change to scout after a duration of time. An improvement made is during employed foraging, if a higher dens location is detected while following the determined Path Integration (PI) it would rather forage that location. PI is the process where a bee continuously changes its baring vector towards its starting location to be able to share the direction other bees during the dance [Jade 22].

2.4 Prioritized Clustering / Foraging

An environment containing prioritized and non-prioritized object throughout may introduce the following: It may occur that prioritized items are encircled or obstructed by non-prioritized objects. There are two methods to resolve this, by foraging the non-prioritized objects or by cluster the which will separate the object types away from one another. To achieve success and optimal result for the multi-problem both of these solutions is used.

3 Algorithm Description

3.1 Desert-ant cemetery clustering

The ant cemetery clustering algorithm is dependent on the densities of an ants immediate area. A laden ant will drop its payload if the density surpasses the probability barrier. During the scouting process ants will move towards the lowest dens area and the highest during clustering. The process and states are described by Figure.1 as follows:

1. A scout searches randomly until a populated area is detected. Upon object detection scouts move towards the lowest dens area populated with objects.

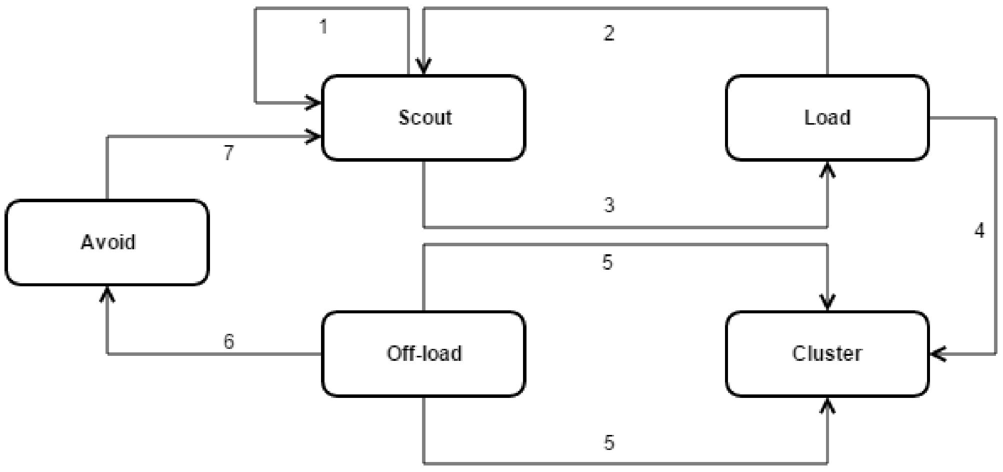


Figure 1: Desert Ant Cemetery Clustering State Diagram

2. An ant will attempt pick-up of an object once arrived at the source area. The pick-up probability is calculated with the density detected from the surrounding objects. If the probability surpasses ϕ it will be picked up.
3. Otherwise it will continue to follow the density and attempts to find an object that has a low enough density around it.
4. Once laden an ant changes its detection for the most dens area within its perceived vicinity.
5. The same probability calculation is used by scouting, the object will be dropped when it is above μ for successful clustering.
6. After the drop ants avoid the area for the max of its detection range, to avoid the possibility of being caught optimizing a single cluster instead of clustering outliers.
7. Finally ants resorts to the default scout method.

3.2 Honey-bee Foraging

The honey bee algorithm consists of two type of robots and initialized by random as unemployed foragers or scouts. The position of each robot is initialized along the starting column of the grid for an increased spread of robots. A sink where objects are drop is not limited to a single location for all robots but the initialization position of the specific robot is its own drop point. All robots states are depicted in Figure.2 and described as follows:

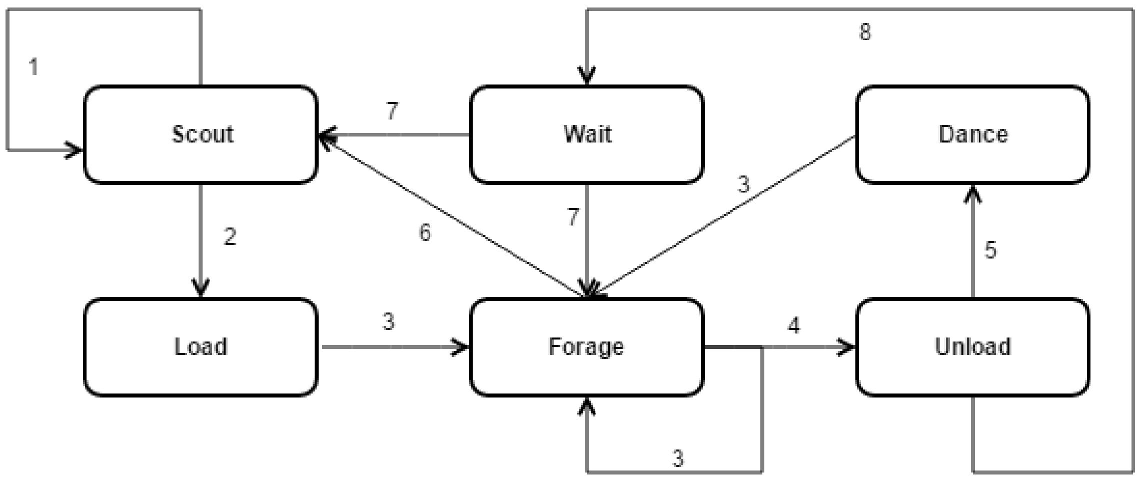


Figure 2: Honey Bee Foraging State Diagram

1. A bee in the scouting state does a random walk through the environment looking for the area with it highest density of objects.
2. Once a dens area is detected and the probability of pick-up is greater than ϕ an object will be loaded and the bee will start foraging.
3. As laden the bee forages the object along its PI back to the drop sink where it will drop the object. If not laden the bee follows the direction vector back towards the previous found location for the measured distance. During the return process a higher dens area may be detected where upon the bee will changes its bearing vector, this is done if the

density is greater than 1.4 of the previous determined density.

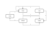
4. An object is unloaded from a laden bee once it reaches its sink and deposits it. If the bee was in the waiting state before foraging it will return to wait else as an unemployed forager it will continue to forage the site.
5. After a successful deposit the bee calculates the probability of if the found site is successful. If so it will perform a dance for bee in a close vicinity to promote the sites PI as well as it distance otherwise the robot will do a solo forage.
6. An employed forager may abort after no object where able to be found where upon it will return to the sink and go into a waiting state.
7. A bee in the waiting state will be employed until it detects a dance within its vicinity that is deems profitable. The bee may also resort to unemployed scouting if it does not detect a dance for a prolonged time $tmax$.
8. A bee that was in a waiting state before being an employed forager will return a waiting state once it off-loads.

4 Simulation Configuration

The simulated environment is representation with a 2D grid which has values assigned for every state for robots and the type of object and their own state of carried. Each object and robot fills one index of the grid. Objects can be picked up by robots once on top of an object or move around it as an obstacle to get to higher dens areas. Where an area is calculated over a 5 index range around the robots position.

An object is successfully foraged once it reaches the end of the environment at column zero where object will be dropped and removed from the environment. The adoption to the setup for a mining environment where the sinks will be place at specific locations would be simple. Different environment layouts are used for testing, with this four object distributions: Uniform

(Fig 3.a), Clustered with Lumer-Faieta (Fig 3.b), Vein (Fig 3.c), Gaussian (Fig 3.d).


(a) Uniform

(b) Clustered

(c) A Vein

(d) Gaussian

Figure 3: Environment Scatters

The configuration are setup as follows where each is configuration is simulated 30 times. Scatter layout, the placement of object as names above to test different dispersion resulting from an explosions. Grid sizes representing the number of position units increasing complexity and required simulation time, $S = 50, 100, 200$. The amount of robots used to cluster and forage objects, $c = 10, 30, 50, 70, 100$. Grid coverage as a percentage of objects placed against open locations for movement, $p = 5\%, 20\%, 50\%, 70\%, 90\%$. A ratio representation, $r = 0.2, 0.25, 0.333, 0.5, 0.667, 0.75, 0.8, 1$ as the number of prioritized objects against non-prioritized object. A priority value is used to determine the priority of a populated area calculated per item detected for $\tau = 0.2, 0.25, 0.333, 0.5, 0.667, 0.75, 0.8, 1$. Honey-bee specific values are used for the allowed waiting time of $tmax = 100$, the max time spent foraging is determine by 1.3 times of the distance acquired from the dance.

A stagnation condition has been created to prevent endless random walks that is determined from when the last pick up occurred. This stagnation limit is set at 400 time the grid size which is applied to the clustering and foraging processes. Also a 100000 limit on iterations across both pre-clustering is required and foraging combined.

5 Results

The result will use the priority of item, ratio of items and the iteration count aimed at solving the following hypotheses with regards to the percentage of items foraged:

1. The relative effect of stagnation and set limitation across the algorithms.
2. Pre-clustering will increase the success rate that more priority items are foraged against only performing foraging.
3. Pre-clustering will require a higher iteration count but will be acceptable due to a significantly higher success rate.

Section 5.1 to 5.3 will cover each hypotheses independently, attempting to solve each through statistical results of the simulation of above.

5.1 Relation between Performance and Stopping Conditions

This section aim to establish the time impact that pre-clustering has the process against only foraging. Due to the constraint on stagnation as well as iteration count causes that an amount of simulation are preemptively stopped. As from the percentages of stopped simulations shown in Table 5:

	Stagnated	Limited	Average Iterations
Non-Cluster	20.95	0.0	36073
Pre-Cluster	31.06	35.5	77578

Table 1: Stopping Condition Limitations as Percentages

These statistics are recorded over the entire simulation, including the pre-clustering process when it is used. This indicates that the clustering consumes a high amount of iteration from the 2.15 increase on iteration count.

The stagnation rate is based on the amount of iterations robots spend unladen, it is preemptively stopped once the above set limit is passed. As these statistics show 66.56% of the the pre-cluster simulations where preemptively stopped where only 20.95% of non-cluster was affected. Although this has hindered the performance of the simulations it has still performed exceptionally.

5.2 Relation between Item Priority and Ratio of Items Analysis with Regards to Foraging Success

The relation analysis of r and τ is describe in Table 2 as a representation of how successful the simulations where. These values were calculated in terms of the percentage of items foraged against the amount of priority items placed. A condensate form is shown in Table 1.

	Average	Std Dev
Non-Cluster	0.884	0.279
Pre-Cluster	0.740	0.361

Table 2: Performance Summation

As from shown the pre-cluster as been slightly less successful then the Honey Bee foraging alone. Although the pre-clustering has still obtained a higher success rate for dynamic environments as tested by Jade [Jade]. Also improvements made on the implementation [Jade] has shown a 7.7% increase on success rate and also a 1.5% lower standard deviation for the Honey-Bee foraging without clustering. The detailed overview of the analysis is giving in Table 3.

The pre-clustering algorithm has high success points but also due to the limitations discussed above low point creep in. This how ever does not hinder the performance showing pre-clustering does in fact show promise and the ability to increase the overall success rate of the foraging process.

5.3 Relation between Item Priority and Ratio of Items Analysis with Regards to Duration

This section uses the amount of iterations against the success rate achieved during the simulation. The higher the success rate the lower the shown value and the inverse is the true for the

Algorithm	r	t							
		0.2	0.25	0.33	0.5	0.667	0.75	0.8	1.0
Non-Cluster	0.2	0.700	0.540	0.898	0.799	**0.968**	0.893	**0.941**	**0.998**
	0.25	**0.999**	**0.999**	**1.000**	**1.000**	0.968	**1.000**	**0.950**	0.697
	0.33	0.554	0.899	0.819	**0.961**	**0.904**	**0.938**	**0.901**	0.561
	0.5	0.792	0.712	**0.954**	0.885	**0.992**	**0.941**	**0.994**	**1.000**
	0.667	**1.000**	**1.000**	**1.000**	**0.968**	**0.950**	0.900	0.576	0.800
	0.75	0.730	**0.951**	0.894	**0.995**	**0.939**	0.706	0.571	0.895
	0.8	0.826	**0.975**	0.893	0.942	**0.999**	**0.999**	**1.000**	**1.000**
	1.0	**1.000**	**0.968**	**1.00**	**0.950**	0.717	0.561	**0.900**	0.810
Pre-Cluster	0.2	0.690	0.488	**0.890**	0.690	**0.960**	0.680	**0.890**	0.756
	0.25	0.870	**0.995**	0.722	**0.914**	0.632	**0.948**	0.564	0.716
	0.33	0.528	**0.899**	0.684	**0.814**	0.466	0.448	0.469	0.542
	0.5	0.756	0.638	**0.906**	0.710	**0.968**	0.652	**0.998**	0.869
	0.667	**1.000**	0.637	**1.000**	0.579	0.884	0.590	0.574	0.762
	0.75	0.674	**0.837**	0.656	0.503	0.284	0.706	0.506	**0.898**
	0.8	0.695	**0.956**	0.682	**0.890**	0.756	**0.876**	**0.997**	0.723
	1.0	**0.916**	0.632	**0.947**	0.564	0.731	0.534	**0.905**	0.716

Table 3: Completion Based Performance

iteration count. Table 4 shows a summation of the process time verses its success rates.

	Average	Std Dev
Non-Cluster	0.042	0.421
Pre-Cluster	0.268	0.311

Table 4: Time-Performance Summation

As Table 4 shows the pre-clustering is a not every efficient regarding item foraging per iteration. It proves to have be 6 times less efficient per iteration then only foraging with regards to time constraints. Although it has a lower standard deviation resulting in a more consistent result. This may be due to the high amount of simulations being limited by iterations.

Algorithm	r	t							
		0.2	0.25	0.33	0.5	0.667	0.75	0.8	1.0
Non-Cluster	0.2	**0.057**	0.180	**0.0567**	0.338	0.146	0.454	0.325	0.500
	0.25	0.253	**0.123**	0.492	0.177	0.563	0.180	0.672	**0.052**
	0.33	0.190	**0.081**	0.422	0.140	0.544	0.326	0.504	**0.065**
	0.5	0.166	0.253	0.192	0.440	0.208	0.632	**0.073**	0.286
	0.667	**0.102**	0.522	0.142	0.564	0.308	0.482	**0.066**	0.170
	0.75	0.319	0.198	0.478	0.226	0.642	**0.058**	0.188	**0.057**
	0.8	0.352	**0.151**	0.447	0.317	0.494	0.250	**0.120**	0.491
	1.0	0.181	0.564	0.182	0.671	0.059	0.179	**0.062**	0.373
Pre-Cluster	0.2	**0.359**	0.403	0.552	0.534	0.683	0.473	0.700	0.476
	0.25	0.745	0.676	0.695	0.579	0.616	0.606	0.557	**0.381**
	0.33	0.441	0.565	0.550	0.588	0.352	0.355	**0.235**	0.411
	0.5	**0.409**	0.574	0.552	0.656	0.612	0.611	0.612	0.815
	0.667	0.695	0.512	0.765	0.386	0.723	**0.362**	0.437	0.425
	0.75	0.610	0.518	0.608	**0.320**	0.262	**0.370**	0.426	0.560
	0.8	0.540	0.680	**0.474**	0.701	**0.476**	0.754	0.677	0.696
	1.0	0.582	0.615	0.606	0.557	**0.387**	0.448	0.566	0.566

Table 5: Time Based Performance

6 Conclusion

Conclusion

References and Notes