Extending Locust Swarms through Dedicated Scouting Particles

Jacob Zeransky

Abstract – **Convergence to a non-global optimum can be a problem for basic search techniques when working in non-globally convex search spaces with multiple optima. Locusts swarms are a technique explicitly designed to tackle this problem, using "smart" start points to scout for promising new areas of the search space around the previous local optimum. Having dedicated scouts allows for continuous searching of new areas not limited to that around a local optimum, and opens the door to new techniques based solely around scouting behavior. These scouts relay information about their position to their swarm, where it is added to a priority queue based on its value. When swarms "devour" the area around these positions, they also maintain their own local optimum, which gets queued if it is far enough from their previous optimum after they have finished with it. This technique allows swarms that are determining a local optimum to always have knowledge of areas where other optima may exist.**

I. Introduction

Convergence is not always the goal of optimization algorithms when multiple optima may exist in the search space. These non-globally convex spaces can pose a challenge for algorithms not specifically designed for them.

Globally convex search spaces are defined by a single optimum that can be reached by moving through progressively better optima until convergence is achieved. Rosenbrock, Sphere, Booth's, and Beale's functions are all examples that produce a globally convex search space in many dimensions. They are frequently used to test optimization functions such as this one, in areas like precision and velocity of convergence.

The basic particle swarm optimization (PSO) functions well under these conditions [2], but in a multi-modal environment it may never converge the global optimum. A special class of optimization algorithm is needed for these types of search spaces, one which can determine an optimum and then escape it in order to determine the others.

II. Extended Locust Swarms

- All Particles
  - Maintain a best-found optimum
  - Return best-found optimum after desired iterations, or other condition
- Scout Particles 1-S
  - Create S scouts at a randomly generated point
  - Begin searching according to defined behavior
  - Decay previously relayed optimum value if not currently determining optimum in search path
  - When local optimum in search path found, relay position to swarm S
- Swarms 1-S
  (1) Create N particles at random points and randomly assign to swarms 1-S
  (2) Swarm T: maintain a priority queue of locations of interest relayed by scout T, prioritized by value of point
  (3) Particles in T: move towards first location until all are within range, then start iteration counter
  (4) After a set number of iterations near first location, fan out and repeat
  (5) After a second set number of iterations, de-queue first location
  (6) Add best point found by any particle in T to swarm T's priority queue if it is a sufficient distance from the de-queued point and has not already been used by swarm T
  (7) Clear best point found for all particles in T, repeat from step (3)

This algorithm is built on Locust Swarms[1], which are built on WoSP (Waves of Swarm Particles). The largely random re-scattering of particles is modified in the hopes of directing them to promising new areas of the search space at fixed time intervals.

Where WoSP basically uses random directions for the particles and random time intervals, Locust Swarms use the best points from a randomly generated pool around the previous optimum, and only deploys these "scouts" once the best point has been optimized. This behavior is extended in steps (4) to (6), where the fan out allows the swarm to determine if the current location of interest is the optimum and points them in its direction if not. Once the optimum has been found, (6) will fail to queue a new point and the swarm will move on to the next promising location relayed to it by its scout.

The scouts themselves have their own algorithm to define search behavior and relaying conditions. The search pattern is more or less predefined, with scouts rotating out from the random start point in identical paths, however this could be modified to follow the peaks and valleys of the test function. As a scout travels it will determine the optimum of the path it has taken and relay that information to its swarm, if the value of locations it passed through are not better than what it relayed, the scout will decay that value so as to make it more sensitive to the topology of the function.

The continuous scouting also allows swarms to redeploy to more promising locations before the local optimum has been found, if the new location already has a better value than the

local optimum so far. Allowing swarms to search the most promising locations first, theoretically decreases overall search time because they will not waste time determining sub-optimal optima. For example, if a two dimensional test function produced a long valley that lead to a mediocre optimum, the swarm could waste a great deal of time determining that optimum when the global optimum is just over the rise. The swarm's scout could find this location and pull the swarm out of the time sink it was progressing through. However, if this does not prove the case and the global optimum was indeed in the valley, because locations are not de-queued until they have been sufficiently searched, the swarm can pick up where it left off.

The algorithm will continue searching for optima until its iteration limit has been found, however since the search space is limited it can reach a point where there is nothing left to search and simply idle.. In the NetLogo implementation because the global optimum is known beforehand, the algorithm terminates once it has reached it.

III. Results

This extension of Locust Swarms has only been tested in a very limited NetLogo search space. The world uses patches to represent points which severely limits the ability to simulate functions that involve multiple decimal places. As well, the world is limited to two dimensions so the algorithm was designed specifically to function in this environment; addition or reduction of dimensions would require modification of the code.

TABLE I
Convergence on Rosenbrock Optimum

Search Technique	Mean	Unable to Locate Optimum
PSO	53	6
ExLS	657	31

Results for basic PSO and extended Locust Swarms on a 2-dimensional Rosenbrock function, mapped to a 401x401 continuous NetLogo world. 100 trials for each are performed with an iteration limit of 1000, mean and standard deviation are calculated for trials that converged to the optimal solution, and the number unable reach the optimum are counted.

$$fm(x)=\sum_{k=1}^{d}\begin{cases}100-|x_k|, if\ |x_k|>50\\ |x_k|, otherwise\end{cases}$$

The test function fm(x) was designed specifically for testing the algorithms ability to determine and escape from local optima. The function produces three local optima and one global optimum, all of which are nearly identical in their value as well as their surrounding area. When testing this algorithm, swarms can appear to get stuck in an optimum because their scouts have not relayed any locations with higher priority, however they will eventually escape. As shown in table II, all 100 trials manage to determine the global optimum, while the basic PSO only managed to locate it in 23 trials, largely due to the randomized start locations being in close enough proximity to it.

TABLE II
Convergence on Test Function Optimum

Search Technique	Mean	Unable to Locate Optimum
PSO	35	77
ExLS	780	0

Results for basic PSO and extended Locust Swarms on a 2-dimensional specially designed test function, mapped to a 401x401 continuous NetLogo world. 100 trials for each are performed with an iteration limit of 2500, see Table I for further description of presented data.

IV. Parameter Analysis

TABLE III
Analysis on S parameter, N = 25

S	Mean	Standard Deviation
2	827	559
4	811	554
7	726	524
15	725	552

Results for extended Locust Swarms on test function, mapped to a 401x401 continuous NetLogo world with different values of S used for creating scout population, with constant N parameter. See Table I for further description of presented data.

TABLE IV
Analysis on N parameter, S = 3

N	Mean	Standard Deviation
25	829	610
50	773	526
75	770	522
115	793	567

Results for extended Locust Swarms on test function, mapped to a 401x401 continuous NetLogo world with different values of N used for creating swarm population, with constant S parameter. See Table I for further description of presented data.

TABLE V
Analysis on Scouting Technique, S = 3 N = 25

Scouting Speed	Mean	Standard Deviation
1	813	563
3	377	259
7	347	327
15	291	329

Results for extended Locust Swarms on test function, mapped to a 401x401 continuous NetLogo world with different speeds used for scouts, with constant S and N parameter. See Table I for further description of presented data.

The use of dedicated scouts allows different techniques to be subbed in to this part of the algorithm independent of the rest. This implementation only uses one search technique so the only parameter to vary is the scout's speed, but it does have positive results. If even this basic random scouting technique can help the swarms locate local optima, a more extensive and tailored technique could have a significant impact on search time. For example, one involving gradient ascent and the application of a gravity vector to push scouts away from previous relayed points as well as those of other scouts sounds promising. The fact that varying other parameters had little effect on the running time shows they are of little consequence for this testing environment.

V. Conclusions / Future Work

Locust Swarms are a promising new multi-modal search technique that ensures swarms do not become trapped in local optima. By adding dedicated scouts the algorithm can redeploy swarms to more promising locations and cut down on time spent determining sub-optimal optima. This implementation provides a very basic framework for the algorithm where new and better techniques can replace existing ones. Much work has already been done in the area of swarm optimization which pertains to half of the algorithm, so the area of interest should the search technique. As previously stated, gradient ascent and gravity vectors to drive scouts away from already optimized locations would be a good start. There may also be an optimum size of swarm per scout, and a sort of balancing between swarms over queue size could produce better results. Ultimately, the implementation shows it can determine multiple optima so the next step should be more extensive testing on functions of higher dimension to determine its usefulness and the areas that need the most optimizing.

References

[1] Chen, Stephen. "Locust swarms: a new multi-optima search technique." In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pp. 1745-1752. IEEE Press, 2009.

[2] J. Kennedy and R. C. Eberhart, “Particle Swarm Optimization”, in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.